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

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

Winter Semester 2022/2023

August 12, 2022 – March 5, 2023

Munich, Germany

Editors

Georg Carle, Stephan Günther, Benedikt Jaeger, Leander Seidlitz

Publisher

Chair of Network Architectures and Services

Network Architectures and Services NET 2023-06-1

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

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

Winter Semester 2022/2023

Munich, August 12, 2022 – March 5, 2023

Editors: Georg Carle, Stephan Günther, Benedikt Jaeger, Leander Seidlitz

Network Architectures
and Services
NET 2023-06-1

Proceedings of the Seminar
Innovative Internet Technologies and Mobile Communications (IITM)
Winter Semester 2022/2023

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/`

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

Cataloging-in-Publication Data

Seminar IITM WS 22/23
Proceedings of the Seminar Innovative Internet Technologies and Mobile Communications (IITM)
Munich, Germany, August 12, 2022 – March 5, 2023
ISBN: 978-3-937201-77-1

ISSN: 1868-2634 (print)
ISSN: 1868-2642 (electronic)
DOI: 10.2313/NET-2023-06-1
Innovative Internet Technologies and Mobile Communications (IITM) NET 2023-06-1
Series Editor: Georg Carle, Technical University of Munich, Germany

Preface

We are pleased to present to you the proceedings of the Seminar Innovative Internet Technologies and Mobile Communications (IITM) during the Winter Semester 2022/2023. Each semester, the seminar takes place in two different ways: once as a block seminar during the semester break and once in the course of the semester. Both seminars share the same contents and differ only in their duration.

In the context of the seminar, each student individually works on a relevant topic in the domain of computer networks, supervised by one or more advisors. Advisors are staff members working at the Chair of Network Architectures and Services at the Technical University of Munich. As part of the seminar, the students write a scientific paper about their topic and afterward present the results to the other course participants. To improve the quality of the papers, we conduct a peer review process in which each paper is reviewed by at least two other seminar participants and the advisors.

Among all participants of each seminar, we award one with the *Best Paper Award.* For this semester, the awards were given to Andreas Kramer with the paper *Recent Advancements in Privacy Preserving Network Layer Approaches* and Désirée Rentz with the paper *Mechanisms and Protocols for Reliable Communication Networks* .

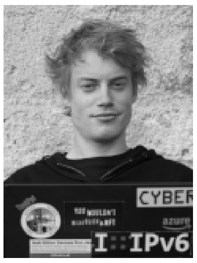
We hope that you appreciate the contributions of these seminars. If you are interested in further information about our work, please visit our homepage `https://net.in.tum.de`.

Munich, June 2023

Georg Carle

Stephan Günther

Benedikt Jaeger


Leander Seidlitz

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
Leander Seidlitz, Technical University of Munich, Germany

Advisors

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

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

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

Eric Hauser (hauser@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

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

Christoph Schwarzenberg (schwarzenberg@net.in.tum.de)
Technical University of Munich

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

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

Henning Stubbe (stubbe@net.in.tum.de)
Technical University of Munich

Florian Wiedner (wiedner@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/ws2223/seminars/`

Contents

Block Seminar

Elaboration of Information and Content Centric Networking . . . 1
Izzet Cetinkaya (Advisor: Markus Sosnowski)

A Survey about the Work of the Coding for Efficient Network Communications Research Group (NWCRG) . . . 7
Dorit Hübner (Advisor: Henning Stubbe, Kilian Holzinger)

Lost in Simulation: Route Property in Mininet . . . 13
Philipp Kern (Advisor: Henning Stubbe, Kilian Holzinger)

Recent Advancements in Privacy Preserving Network Layer Approaches . . . 19
Andreas Kramer (Advisor: Filip Rezabek, Richard von Seck)

P4 16: Out of the Loop . . . 25
Felix Sandmair (Advisor: Henning Stubbe, Eric Hauser)

Seminar

TSN Qbv and Schedule Generation Approaches . . . 29
Vishavjeet Ghotra (Advisor: Max Helm, Benedikt Jaeger)

Reproducible Network Experiments using NixOS . . . 35
Michael Hackl (Advisor: Kilian Holzinger, Henning Stubbe)

Increase the Latency: Emulation Approaches for Real Network Conditions . . . 41
Lukas Haneberg (Advisor: Eric Hauser, Sebastian Gallenmüller)

The IEEE 802.11ad Standard: Challenges and Design Adaptations . . . 47
Mahdi Koubaa (Advisor: Jonas Andre, Stephan Günther)

Combining Machine Learning With Back-Pressure-Based Routing . . . 53
Pauline Laßmann (Advisor: Christoph Schwarzenberg, Florian Wiedner)

Mechanisms and Protocols for Reliable Communication Networks . . . 59
Désirée Rentz (Advisor: Kilian Holzinger, Henning Stubbe)

Survey on the Chinese Governments Censorship Mechanisms . . . 65
Raphael Stadler (Advisor: Lion Steger)

Elaboration of Information and Content Centric Networking

Izzet Fatih Cetinkaya, Markus Sosnowski*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: izzetfatih.cetinkaya@tum.de, sosnowski@net.in.tum.de

Abstract—**Information-centric networking (ICN) and Content-centric networking (CCN) are new architectures that aim to replace or initially support current internet protocols (IPs). Consumers no longer need to wait for a response from hosts to access information with the usage of internet protocols. ICN, speciall through CCN, resolves this waiting time by allowing the data exchange within the naming content.Based on the naming content CCN additionally offers safer architecture for consumers by using cryptography in searched content.In this paper we aim to explain the backstory and the primary working structure of ICN/CCN. Furthermore, we give a small example of an existing tool based on CCN working principles which also aims to explain how CCN can operate and compare CCN and IP in three categories.**

Index Terms—**content-centric networking, information-centric networking, forwarding,internet protocols.**

1. Introduction

The current internet protocols (IPs) ,due to security reasons and slowleness caused by host to host information exchange, are no longer sufficient enough to meet today's demand to reach information faster. Therefore we needed new protocols which could replace IPs or change the way they are operating. In classical IPs, information location and how it is distributed to consumers are the key elements of data distribution. Hence, we need to rely on pipelines and local hosts more than the content we are looking for. However, at the beginning of 2009, researchers in Pao Alto, United States, created another networking architecture that can answer the need for fast and secure information more efficiently [1]. The new architecture is called information-centric networking (ICN). The difference between IPs and ICNs is heavily based on the concept of their working mechanism. While IPs are relying on the host information and *how* they are shared with consumers, in contrast, ICNs are based on the name of the content and *what* is delivered to consumers [1]. Thus, ICN allows reaching information much faster and more secure than IPs thanks to avoidance of host data, which can contain malicisous threads, and caching information to reuse later [2] [1]. This new information-based architecture consists of different sub-structures and sub-architectures. The most used among the new ICN is content-centric networking (CCN). The reason CCN is a more common approach than others is that CCN has an effective way of data exchange. In other words, CCN enables the information exchange between users only on the content that they are looking for. Hence, information flow does not depend heavily on the layer protocols. In this paper, we are going to explain the back story of ICN and CCN in Section 1 briefly, and then we are going to elaborate on the current working structures of CCN with detailed information in section 2. Furthermore, we are going to give an example of a tool that aims to explain how CCN works and in the last section, we are going to compare IPs and CCN in three different categories, where CCN is offering consumers better usage.

2. Background of ICN and CCN

ICN was created to accelerate current internet protocols more efficiently and securely. Hence ICN offers a shift from "Host Centric Network" to "Information-Centric Network." [3]. What does this shift mean? The classical approach of internet protocols (IPs) is based on the host data, which is essential to know where the data storage, that was restricted into four layers; Application Layer (HTTP), Transport Layer (TCP), Network Layer (ISO 7498/4) and Link Layer (Ethernet). However, in ICN, Host-data plays an insignificant role in determining requested information. In contrast, the ICN approach is more in which data has been requested from the consumers. Therefore we can eliminate the long wait time, which IPs are based on, with the help of information-centric networking.ICN has several approaches like Named Data Networking (NDN) [4] or Data-Oriented Networking Architecture (DONA) [5]. NDN is Content-Centric Network (CCN) based architecture. What we mean by based architecture? When NDN is first used, the working priciples and structure were build upon the CCN's working structure [4]. For example, NDN also uses interest and data packets for information exchange,for a more detailed explanation of interest packets (section 3). DONA is, on the other hand, created to accelarete existing application and try to improve security. DONA's working principles aim to change domain name system (DNS) names with flat names, self-certified names (section 3), and DNS name resolution with name-based anycast primitive that lays above IP layer [5]. DONA also improves data retrieval by providing more coherent support for persistence, authentication, and availability [5]. But, CCN is still one of the most common approaches of ICN architecture. While the conception of data request switched to content based rather than the host location, CCN has significant advantages in compare to DONA, to answering the need for ICN by replacing IP Layers.The reason is CCN is the key and common approach of ICN that CCN is more adaptable to new environments and being able to provide better caching and naming in multicast traffic in compare to DONA and NDN [2]. Thus, CCN has the upper hand in comparison to other approaches like Named Data Networking (NDN) [4] or Data-Oriented Networking Architecture (DONA) [5]. Additionally, in some cases, CCN has been considered that is not precisely replacing layer 3 (ISO 7498/4) but plays an additional role in speeding up the resulting process [1].

3. Basic Concept of CCN

As we briefly explained in section 2, ICN based architecture CCN can be more efficient approach to exchange information between providers and consumers. In this section, we elaborate on this efficient approach, which can create switch from IP to CCN with a clear concept of CCN. In order to point out the working mechanism and structure of CCN, the main part of our illustration of structure is based on the new working principles of CCN, which is information exchange within packets or, in other words, faces. CCN data exchange consists of two main bodies, and these are *interest packets* and *data packets.* The interest packet consists of three main layers, which are *content name selector,* and *nonce* [1]. The data packet, however, consists of four layers, those are *content name, signature, signed info* and *data* [1] [2]. As we can clearly identify, the key indicator for both packets is the content's name. Even though both packages contain the same content name as the vital unifier for packets, their core roles are not overlapping but rather complete each other. In other words, the interest packet's name contains the requested information by consumers or CCN nodes, and data packets have the corresponding information from servers or hosts [2].

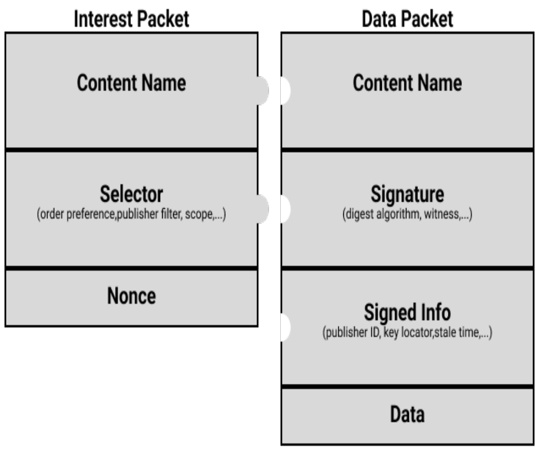

Interest Packet
Data Packet
Content Name
Content Name
Selector
(order preference,publisher filter, scope,...)
Signature
(digest algorithm, witness,...)
Nonce
Signed Info
(publisher ID, key locator,stale time,...)
Data

Figure 1: CCN Package Layers [1]

In figure 1 and figure 2 are visualising the interest and data packets.After visualisation of the packets, we can touch the content of the each layer. The content name layer contains a sequence of name components [4].The signature is defined in two different parts; firt one is signature-info, which indicates digitial signature algorithm and relavent information in local certificate.Second one is signature-value that hold the bits of the signature [4]. Last but not least Nonce is used to carry randomly generated long byte-stting [4]. Additionally, figure 2 shows IP layer

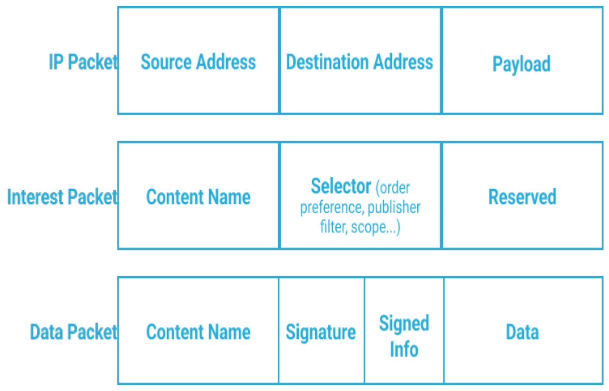


Figure 2: CCN and IP Package Layers Comparison [2]

distribution compared to CCN, which helps us see how content-centric networking differs from current internet protocols.

After we explained the two central bodies of the CCNs, we also want to point out how this mechanism works and how those main strategies support Content Centric Networking.

The main working mechanism that CCN is built upon is the following strategies:

3.1. *Forwarding*

The forwarding strategy is one of the key elements of CCN to transfer data from sender to receiver or receiver to sender. Forwarding has three essential sub-structures:

- Pending Interest Table (PIT)
- Forwarding Information Base (FIB)
- Content Store (CS)

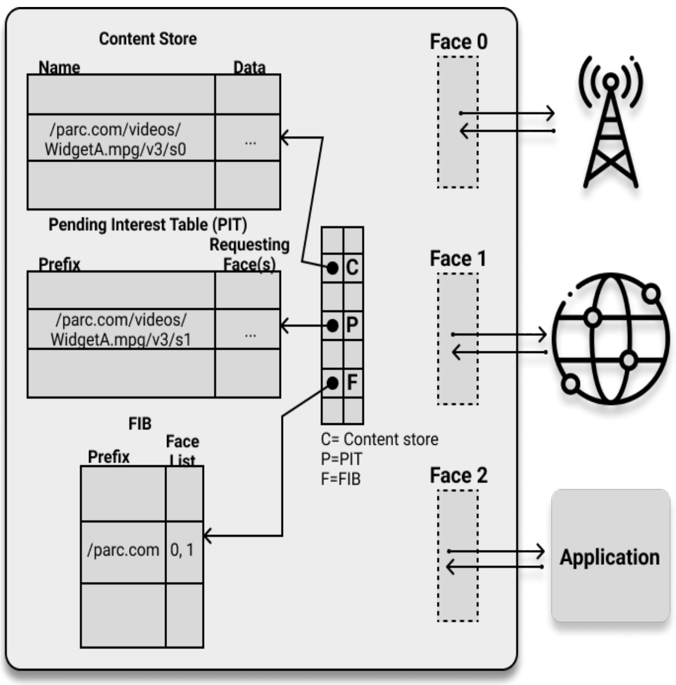
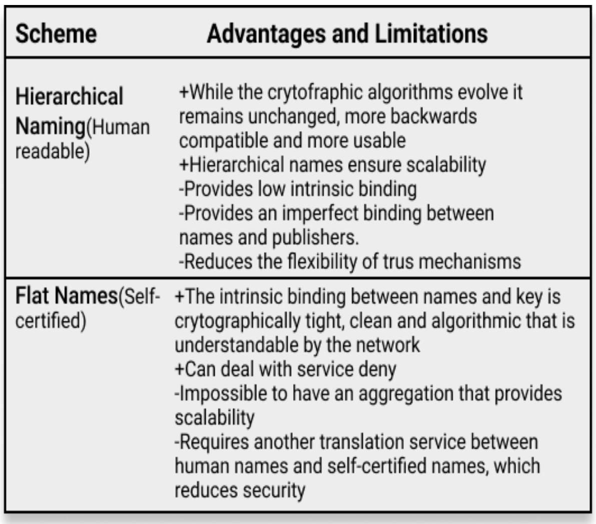


Figure 3: Forwarding structure [1]

In figure 3 it can be seen that each of the three sub-structures works alone or with other structures. In the first step of cooperation work, requests are created by the receivers as following interest packets sent to corresponding faces (PIT). In the second step, FIB transfers the information to potential matching data, which will collect the related data and send it back to the receiver, and in the mean time, CS stores founding data in its storage in order to create a faster response in the future. In [1] the whole process and detailed explanation of each step can be seen better.

3.2. *Naming*

The main concept of Content-Centric architecture is naming the content directly. “Then, publishers place it in the network where it is replicated in caches” [2]. Thanks to the naming strategy, it allows publishers and receivers to have a more secure, more flexible, and more scaleable CCN. In [6] Ghodsi et al. explained how the CCN Naming works. In our structural explanation, we want to also point out two main approaches to Naming. These are hierarchical naming and flat naming. Although they are both responding to the naming process of CCN, they have slight differences. While Hierarchical naming ensures that the content name can be human readable, Flat naming provides self-certified naming. In figure 4 the main differences can be identified better, in order to understand which of the naming approaches are used where. Additionally, figure 5 shows how the structural concept of the two naming approaches works.



Scheme	Advantages and Limitations
Hierarchical Naming(Human readable)	+While the crytofraphic algorithms evolve it remains unchanged, more backwards compatible and more usable +Hierarchical names ensure scalability -Provides low intrinsic binding -Provides an imperfect binding between names and publishers. -Reduces the flexibility of trus mechanisms
Flat Names(Self-certified)	+The intrinsic binding between names and key is crytographically tight, clean and algorithmic that is understandable by the network +Can deal with service deny -Impossible to have an aggregation that provides scalability -Requires another translation service between human names and self-certified names, which reduces security

Figure 4: Naming Comparison [2]

3.3. *Caching*

Caching is also one of the conspicuous structures of CCN. Caching structure is heavily correlated with Forwarding due to the nature of Caching. Caching’s working principle is transferring data from buffer memory, which is holding information temporarily, to cache space memory, where it can be stored for a longer time. This is also to process of Content Store (CS) (see Section A). Thanks to caching, in event of a data request from a consumer, FIB does not require to go to different servers to find

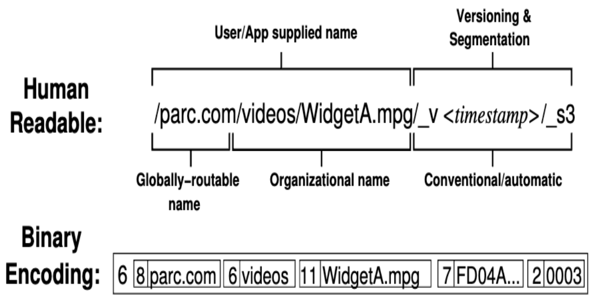

User/App supplied name
Versioning & Segmentation
Human Readable:
/parc.com/videos/WidgetA.mpg/_v <timestamp>/_s3
Globally-routable name
Organizational name
Conventional/automatic
Binary Encoding:
6 8 parc.com 6 videos 11 WidgetA.mpg 7 FD04A... 2 0003

Figure 5: Naming data structure [1]

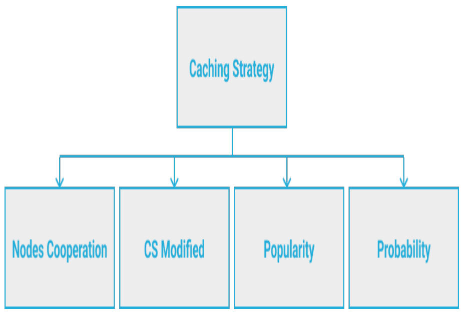


Figure 6: Caching Strategy Layers [2]

information. Instead, it can use the stored data in caching to respond to requests.

In figure 6 various different caching strategies can be seen, which are segmented into four main layers and sublayers.In Nodes cooperation caching, information that currently cached will be shared with neighbour cache [2]. Following cache strategy (CS) suggest that proactive mechanism can be used as an additional modul [2]. In the third CS data packet processing is focused [2]. And in the last one higher cache hit is the key indicator [2].

3.4. *Security Mechanism*

Security structure is based on four main components; those are confidentiality, provenance, integrability, and availability [2]. Thanks to these components, CCN eliminates malicious threads, one of the IP’s main security threats. "In CCN, all content is authenticated with digital signatures, and private content is protected with encryption. This is a critical enabler for CCN’s dynamic content-caching capabilities. If you are to retrieve content from the closest available copy, you must be able to validate the content you get" [1].Hence, CCN enables secure and trustable internet to consumers.

In [1] content-based security more details are explained with experiment. Which helps to elaborate, how CCN can be safer than IPs.

3.5. *Monitoring*

While CCN allows consumers to request data simultaneously, Monitoring needs arise with these simultaneous actions. The main task of Monitoring is enhancing the data that has been transferred to protect CCN from attacks

[2]. The enhancing mechanism works as following; Monitoring captures and analyses the information concerning content distribution flows [2]. In papers [7], [8] different tools have been provided to us for Monitoring.

4. An Existing Tool for CCN

In this section, we want to give a small example of CCN based tool [3], which helps us to understand how CCN works. CONET: A Content-Centric Inter Networking Architecture aims to provide consumers with named data rather than host data [3]. By doing so, CONET either replaces existing IPs with additional noted IPs that make them content aware, or creates new CCN-based networking that can conduct data requests with content-name-based architecture.

In figures 7 and 8, CONET's architecture and primary working packet system can be seen.

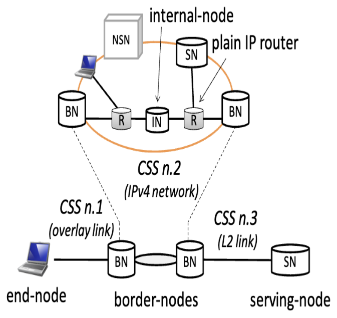


Figure 7: CONET Architecture [3]

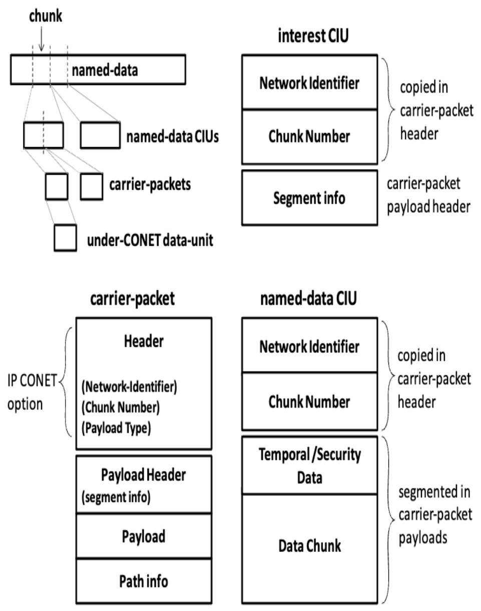


Figure 8: CONET Information Unit [3]

5. Comparison and Discussion

In our last section, we aim to point out three categories where CCN has the upper hand compared to IPs.These are Availability, which we also mentioned (see section D), Security, and Location dependence. We want to start with the one we think, where CCN resolves so many issues compare to IP and this is Security.

- CCN's core structure relies on the data transfer on the content name. Therefore transferred data is no longer a property of pipelines that move data from one another without caring about the content itself but focusing on the data holders. " In CCN, all content is authenticated with digital signatures, and private content is protected with encryption" [1]. Thus, CCN provides a more robust security system for malicious attacks on data transactions. Another defining characteristic of CCN's security is "Key Handling" (Trusting Key). In [1] it is explained under three sub-categories, which are keys directly addressed to the problem, second publishing only one key, and third CCN is not empowering one key for all sizes but it creates trust between publishers (senders) and consumers (receivers).
- The second category that we want to compare CCN with IP is Location dependence. While ICN and CCN are not relying on the host locations and connections between hosts. This independence allows CCN to operate easily without being restricted by internet protocols. Thus, CCN-requested data engages with the consumers faster, and the information is more reliable.
- Last but not least, Availability. CCN's availability is greatly based on the CCN's flexibility. More like location dependency, requested data is not bonded to any host location. Hence, in the case of data requests, contented data can be transferred flexibly to one another. Availability allows CCN to provide more reliable data to consumers.

6. Related work

Content-centric networking is still growing in architecture that is not fully established yet. Therefore there are many ongoing surveys, experiments, and collaborations with other tools. For example, Ahlgren et al. [9] tackle the design choices of ICN and, respectively CCN. In another example, Nakamura et al. [10] focus on how ICN will cooperate in the event of network failure. Even though ICN is offering more sustainable data exchange, it does not entirely eliminate the risk factor of failure, but it decreased significantly. The findings of the [10] indicate thanks to ICN topology, in case of network failure, ICN still operates more efficiently than old internet protocols due to selective node removal. Ghali et al. [11] handle the possible problems with the high usage of CCN. While security is among the key aspects of CCN, this paper should be mentioned in our writing to underline CCN security.

7. Conclusion

In conclusion, we tried to elaborate why ICN and CCN are replacing or accelerating current Internet Protocols. Additionally, we elaborate on the current state of CCN and how CCN is structured. By doing so, we briefly mentioned five critical structures of the CCN. So we could understand the CCN process better. Furthermore, we give an example of a tool that operates under CCN protocols and underline three aspects where CCN works better than IP. As a result, we can say that CCN has significant advantages in security, flexibility, and mobility, which can make CCN more reliable and more demanded in the near future.

References

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, “Networking named content,” in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT ’09. New York, NY, USA: Association for Computing Machinery, 2009, p. 1–12. [Online]. Available: https://doi.org/10.1145/1658939.1658941

[2] R. Jmal and L. Chaari Fourati, “Content-centric networking management based on software defined networks: Survey,” *IEEE Transactions on Network and Service Management*, vol. 14, no. 4, pp. 1128–1142, 2017.

[3] A. Detti, N. Blefari Melazzi, S. Salsano, and M. Pomposini, “Conet: A content centric inter-networking architecture,” in *Proceedings of the ACM SIGCOMM Workshop on Information-Centric Networking*, ser. ICN ’11. New York, NY, USA: Association for Computing Machinery, 2011, p. 50–55. [Online]. Available: https://doi.org/10.1145/2018584.2018598

[4] L. Zhang, “Named Data Networking (NDN) Project,” *NDN-0001 October 31, 2010*, vol. 25, no. 26, pp. 1–18, 2010.

[5] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, “A data-oriented (and beyond) network architecture,” vol. 37, no. 4, 2007. [Online]. Available: https://doi.org/10.1145/1282427.1282402

[6] A. Ghodsi, T. Koponen, J. Rajahalme, P. Sarolahti, and S. Shenker, “Naming in content-oriented architectures,” in *Proceedings of the ACM SIGCOMM Workshop on Information-Centric Networking*, ser. ICN ’11. New York, NY, USA: Association for Computing Machinery, 2011, p. 1–6. [Online]. Available: https://doi.org/10.1145/2018584.2018586

[7] W. Kang, B. Sim, J. Kim, E. Paik, and Y. Lee, “A network monitoring tool for ccn,” in *2012 World Telecommunications Congress*, 2012, pp. 1–3.

[8] D. Goergen, T. Cholez, J. François, and T. Engel, “Security monitoring for content-centric networking,” in *Data Privacy Management and Autonomous Spontaneous Security*, R. Di Pietro, J. Herranz, E. Damiani, and R. State, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 274–286.

[9] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, “A survey of information-centric networking,” *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, 2012.

[10] R. Nakamura and N. Kamiyama, “Analysis of content availability at network failure in information-centric networking,” in *2020 16th International Conference on Network and Service Management (CNSM)*, 2020, pp. 1–7.

[11] C. Ghali, G. Tsudik, and E. Uzun, “In content we trust: Network-layer trust in content-centric networking,” *IEEE/ACM Transactions on Networking*, vol. 27, no. 5, pp. 1787–1800, 2019.

A Survey about the Work of the Coding for Efficient Network Communications Research Group (NWCRG)

Dorit Hübner, Henning Stubbe*, Kilian Holzinger*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: dorit.huebner@tum.de, stubbe@net.in.tum.de, holzinger@net.in.tum.de

Abstract—**There has been a lot of research on improving transmission quality in networks. One approach is coding on nodes within the network (NC). This can make transmission more secure, reliable, and quick. A group of scientists, in this field, formed in 2013 to gather knowledge and identify open research questions about NC. This paper surveys the content of their finished work and puts it in context with other literature. We concluded that NWCRG fulfilled their goals by examining multiple different topics in NC, showing research challenges, and furthermore solving one challenge themselves.**

Index Terms—**nwcrg, network coding, satellite communication, congestion control, forward erasure correction, content-centric networking, named data networking**

1. Introduction

Encoding and decoding packets not only at the ends of a transmission but also at intermediate nodes of a network, is the central idea of Network Coding (NC) [1]. Network coding can enhance performance in terms of quality (Sections 2.2, 2.3), quantity (Section 2.2), and security [2]. Fragouli and Soljanin decscribe one typical instance of NC [3]: An intermediate node, receiving two bit flows, can halve the payload by bitwise applying the XOR-operator to these two flows. Then the combination is forwarded as one flow instead of two flows to the next node of the network. The receiver gets sufficient knowledge from other components in the network, to calculate the original data of the two flows. This can be done in multiple ways. One is elaborated in the Subsection Two-Way Relay Channel Mode of Section 2.2.

This paper gives an insight into NC by surveying documents of the *Coding for efficient NetWork Communications Research Group* (NWCRG) and putting them in context with related literature. NWCRG published four RFC documents to fulfill their goals of standardizing NC communication, gathering knowledge about NC applications in practice [2], and encouraging researchers to tackle unsolved challenges. These RFCs cover different topics, give an insight into NC, and are summarized in this paper. A detailed survey like this has not yet been conducted and published, while the results of NWCRG have been cited multiple times ([4], [5], [6]) and an overview of NWCRG as a group has been given [7].

The structure of the paper is as follows: In Section 2 the RFCs and related work RFC are examined. Subsection 2.1 gives an overview of definitions and taxonomies gathered by NWCRG. This is followed, in Subsection 2.2, by detailed suggestions to implement NC in satellite communication systems. Different possible uses of congestion control along with forward erasure correction code, a specific NC code, are described in Subsection 2.3. The last RFC, in Subsection 2.4, deals with NC for information centric networking. Section 3 concludes the paper.

2. Work by NWCRG

NWCRG finished and published four RFCs [8]. These are summarized in the following sections. The focus of the summary lies on analyzing the main concepts for RFC 8975, RFC 9265 and RFC 9273. For RFC 8406, an overview of the structure of the document is given because it mostly contains brief definitions which are summaries themselves.

2.1. Terminologies and Taxonomies in NC (RFC 8406)

Overview. The document RFC 8406 [9] by Adamson et al. gives an overview of terminologies and taxonomies in NC, focusing "on packet transmissions and losses" [9] in non-physical layers. RFC 6726 [10], RFC 6363 [11], RFC 5052 [12], RFC 5740 [13], and RFC 5775 [14] were the main sources for the definitions.

First, "General Definitions and Concepts" [9] are introduced, serving as a detailed glossary for the document. Seventeen key terms and their synonyms (or in some cases words to differentiate from) are introduced by describing them. They can roughly be summarized as terms about erasure, coding, symbols, payloads, packets, nodes, and flows.

Second, a "Taxonomy of Code Uses" [9] is described by differentiating between Source and Channel Coding, Intra- and Inter-Flow Coding, as well as Single- and Multi-Path Coding.

Third, a sketch of different, partially matchable coding types is conducted: Linear Coding along with its variations Random Linear Coding and Adaptive Linear Coding combine input data and coefficients. Block Coding and (Fixed/ Elastic) Sliding Window Coding are mutual alternatives for handling data flows. In Systematic Coding, source data is part of the encoded data. Rateless Coding can produce an unlimited number of different codes of the same source data.

Fourth, basic coding terms are enumerated and defined. Among others, names of different sets of data and their sizes are explained (e.g. Decoding Window, Payload Set). In addition, relevant terms in the context of Linear Codes are mentioned (e.g. Coding Coefficient, Finite Field).

To code in practice, requirements are defined by schemes, needed for a correct coded data transfer. For example, the *Forward Erasure Correction* (FEC) Scheme specifies a distinct FEC code and protocol. FEC describes a type of code, only used in channels that will drop a data packet if it has flaws. In this scheme, it is sent with a protocol, which carries decoding information. One part of the protocol [15], the FEC Payload ID, is described further in the document. It serves as an identifier for segments of a packet which are used in the coding process.

Related Work. Terms and important concepts are generally explained in works about NC ([16], [17]) , but there has not been a document with this focus before RFC 8406, that is widely available. Its content has been reused in the field of NC, mainly by NWCRG members ([16], [18], [19]) but also by others. For instance, Zverev et al. defined some terms based on the RFC 8406, for their work about robust QUIC, a low latency transport protocol [4].

2.2. NC for Satellite Systems (RFC 8975)

The objective of RFC 8975 [19] is, to show opportunities to integrate NC techniques in *SATellite-COMmunication* (SATCOM) networks.

One use case of a SATCOM network is the communication of two devices on the ground over a satellite in space. As the devices can communicate with a satellite, they are called (satellite) terminals. If a direct transfer of data from one terminal to the other is not possible, the sender first transmits the data to the satellite, which forwards it further to the receiver. The transmission usually requires arranging the data in multiple packets, which are sent individually.

In a SATCOM process, NC can be used to reduce the amount of sent packets, decrease the occurrence of mistakes in the transmission and support a quicker recognition along with re-submission of lost packets, while not diminishing the amount of transferred data content.

Two-Way Relay Channel Mode. Using NC, it is possible to reduce the number of packets sent, in a Two-Way Relay Channel Mode. For the continuous communication of two terminals with each other, each terminal transmits a data flow to a satellite. Instead of forwarding the original messages to the receiving terminals, the satellite combines the two data flows. This one flow is then sent. It is received by both terminals as illustrated in Figure 1. By knowing what they sent before, they can decode the data and read the message of the other terminal. This is how, by not sending two data flows but one which carries data for both terminals (multicasting), the amount of data is reduced.

Reliable Multicast. Multicast provides an opportunity to recover lost data, using NC: Packets in a multicast flow are sent from one satellite to multiple terminals. If a packet gets lost at a terminal, the terminal sends a negative acknowledgment back. A repair packet is then encoded in the multicast flow, in a way that that the addition of the repair data does not require sending additional packets. Using this packet, the terminal restores the data. This kind of Reliable Multicast can be used in multicasts or broadcasts described in "Secure Hybrid In Network caching Environment" by the European Space Agency [20], in NACK-oriented reliable multicasts and in file delivery over unidirectional transport [21].

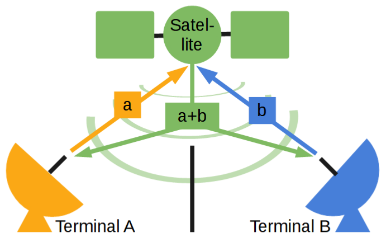


Figure 1: Two terminals communicating in a Two-Way Relay Channel Mode

Hybrid Access. NC application can be used to deal with packet losses in multiple-path communications like Hybrid Access. Using it can also lead to higher flexibility towards the order of packets. NC is applied at the transport layer, more precisely at the at any end user equipment and/ or the concentrator. The concentrator serves as the interface which aggregates multiple channels to the server. This approach has been implemented and published in an ETSI Technical Report [22].

End-to-End Encryption. Packet losses are usually prevented by a *Performance Enhancing Proxy* (PEP) server in a LAN to satellite transmission but User Datagram Protocol based end-to-end encryption makes PEP unusable. Therefore, losses would occur in an end-to-end encrypted wireless LAN SATCOM system. Network Coding may be applied at multiple points during the transmission process – at the end user, satellite gateway, access gateway, or network function. The usage may result in a reduction in packet loss.

Other Packet losses. Sub-second varying physical channel conditions will not necessarily be corrected on the physical layer in time. Consequently, packets get lost. However, they may be recovered through NC mechanisms in other layers.

Another cause of packet losses may be gateway handovers. Reasons for that loss, for instance, flaws in synchronization or trigger-algorithms, can be reduced by using NC.

Research Challenges. In the process of writing the document, the following open research topics were identified (read [19] for more details): Combining NC and Congestion Control, which is used in most SATCOM Systems. Balancing the trade off between benefits of redundant information to recover mistakes and adding too much redundancy in the context of quickly varying channel conditions. Several topics about the implementation of NC in Virtual Network functions. The deployment of Delay/Disruption-Tolerant Networking.

Related Work. Even though there has been little commercial application, a lot of research about NC in SATCOM systems has been done. Cloud et al. encountered challenges when implementing NC in a SATCOM system [23]. Ghanem et al. described two "Channel Virtualization Schemes for Satellite Multicast Communications" [17]. Some of the research cited the RFC 8975 or its draft: Chiti et al. focus on NC applied in SATCOM for reliable multicasts [24]. Thomas et al. reference the RFC 8975 for a description of a generic SATCOM architecture in an article about Google QUIC [5].

2.3. Forward Erasure Correction Coding and Congestion Control in Transport (RFC 9265)

RFC 9265 [25] elaborates different possible relations of *Forward Erasure Correction* (FEC) and transport layer *Congestion Control* (CC). FEC is implemented through NC, processing and changing packets on intermediate nodes of a network. The goal of the FEC code, introduced in Section 2.1, is restoring lost packages at the end of a transmission. The encoding process results in so-called symbols. Specifically, source symbols which contain the source data and repair symbols which contain repair information for one or more source packets. The number of repair symbols is usually decided by a certain rate in comparison to the number of source symbol or by a fixed number. These symbols get reassembled in network packets, which are transmitted. When decoding, the information of the repair symbols is used to restore source symbols that got lost in transmission. A repair symbol can contain the data for exactly one source symbol or a combination of source data. E.g., storing in one repair symbol the XOR combination of multiple source symbols, the repair symbol can restore one source symbol, if exactly one source symbol got lost.

If reliability is not needed, FEC is usually implemented as an alternative to mechanisms for reliable transfer like the retransmissions of lost packages. Because it is possible that a source symbol and its repair symbol get lost, it is used for partially reliable or unreliable data transfers. An example for a fitting protocol is QUIC with the unreliable datagram extension [26]. In special cases a reliable transfer with FEC is also possible (details in [25]).

FEC can be applied right before, within, or after the control entities of the transport layer, as illustrated in Figure 2. This position of FEC determines the possibilities to communicate with CC of the transport. Congestion Control is a mechanism of the sender and the receiver, which calculates if the path is congested and adjusts a congestion window accordingly. By the size of the congestion window, the sender then knows how many packets it can send without losing data by congestion.

FEC above the transport layer. With FEC applied above the transport, the data is FEC encoded before CC and FEC decoded after CC as shown in Figure 2 in green.

CC gets network packets, on which it calculates the congestion. It does not matter if these network packets contain repair or source symbols. Hence, CC can work as it would without FEC and has the same control over the congestion as it would have without.

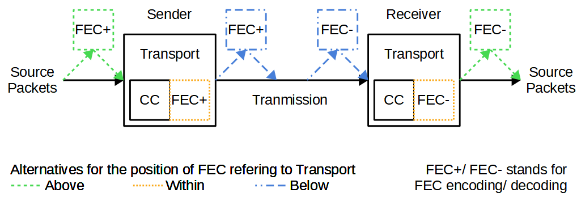


Figure 2: Overview FEC and CC

The core advantage of using FEC at this position lies in the fact that CC can be implemented without special considerations. This might not be the case for FEC within or below the transport, as will be explained.

FEC within the transport layer. The application of FEC within the transport, allows a joint control of CC and FEC. The source packets get encoded by the same controller that decides at what time packets are sent to the receiver. At the receiver a joint controller operates FEC and CC. It can indicate congestion and the use of repair packets to recover source symbols to the sender.

Therefore, it is a flexible solution. The sender controller knows about congestion as well as the number of lost packets and consequently can adjust the number of repair packets to the system's needs.

Should the focus of the system be latency performance, repair packets only get send if no congestion is induced by the additional data.

If there is a lot of data traffic in relation to the transmission capacity and some packets get lost, blocking a fixed percentage of the transmission capacity for repair packets might be useful. A separate CC mechanism can then be implemented for repair symbols, which can be send independent of the source symbol congestion.

The system may dynamically adjust to its current needs by balancing higher congestion and the number of repair packets used in relation to the ones sent. If in relation many repair packets get used, it is likely that the channel is unreliable and transmits a smaller share of packets correctly. To lose as little data as possible more repair packets might be sent, accepting a lower transmission rate. And the other way around if high congestion occurs, the number of repair packets might be reduced, accepting more lost source packets.

A drawback of FEC within transport is that the fitting solutions for the system might require a complex implementation.

FEC below the transport layer. Figure 2illustrates the position of FEC below the transport in blue. Encoding happens after the transport layer forwards data but before the link layer processes it. Communication between FEC and CC is not planned.

This position of FEC is generally beneficial if numerous packets get lost in a transmission. The repair symbols then are employed and lead to a better transmission performance.

The RFC covers only the scenario of the transport controllers not knowing about FEC. FEC sends its repair symbols on top of the original data and restores original packets before they reach the transport receiver, which does congestion control. The receiver misunderstands restored packets which were lost, as sent correctly. Therefore, normal loss-based congestion detection does not work correctly. For instance, a CC controller miscalculates a congested path as good because of FEC repairs the data in time. Using existing signals to indicate a restoration, might be a possibility to still use loss-based congestion. In contrast, delay-based congestion detection works fine. The delay of source packets, which is induced by congestion by source symbols and repair symbols, shows the controller, that a reduction of packets is necessary. This stops congestion from building up. Problems may still occur if, due to external circumstances, congestion generally cannot be prevented. In this case, sending a fixed amount of repair symbols at a fixed rate and applying a separate congestion control entity for repair symbols, might enhance a transmission.

Related Work. Most research in this area investigates which joint dynamic control of CC and FEC is best for a specific system. The following works are examples for this: Tsugawa et al. proposed a general Adaptive FEC Code Control for TCP video streaming in 2007 [27]. In 2017 TCP-TFEC, a method to improve the throughput in wireless LAN, in comparison to Tsugawa's work, was suggested by Teshima et al. [28]. Sharma et al. published work about a multi-path loss-tolerant transport protocol, using CC and FEC [29].

Furthermore, there has been research in NC that has buildt on the RFC 9265, for instance Wu et al. cited RFC 9265 as a reference for complex implementations of FEC and CC in the Transport layer, in their article on A Survey on Multipath Transport Protocols Towards 5G Access Traffic Steering, Switching and Splitting [6].

2.4. NC for Information-Centric Networking (RFC 9273)

RFC 9273 [18] shows the current state of research in NC for *Information-Centric Networking* (ICN), particularly Content-Centric Networking and Named data Networking, by explaining main concepts, technical considerations, and potential challenges.

Process. An ICN can simplified be described as multiple consumers (e.g. end user) connected by multiple forwarders to multiple producers (e.g. server). To receive data, a consumer must create an interest packet, which describes what data it is interested in. Then it sends the interest packet to a so-called forwarder, a node in the network. The forwarder first checks if his cache, also called content storage, contains the requested data. If successful, it sends the data back. Otherwise, searches for a fitting entry in the *Pending Interest Table* (PIT), if it already contains an entry, the two requests can be aggregated. Otherwise, it inserts a new entry, containing the name of the request along with the requesting interface identification. Then it forwards the request to another fitting interface. If the interface is a forwarder, the same process repeats. In contrast, if the interface is a producer, it prepares the wanted source packets by grouping them into blocks. Each block consists of a fixed number (k) source packets, which are encoded with a coding vector, chosen by the producer. The encoded block which contains k source packets and additional repair packets, is sent to the requester. Following the trail of PIT entries, the data is transported back to the consumer. Forwarders passing the data, can store the data in their content storage and can recode repair packets, if they have sufficient knowledge. The consumer is usually able to decode the data after receiving at least k source or repair packets of a block. It also takes security measures, for instance checking for origin authentication.

Technical considerations. Two important technical considerations, mentioned in the RFC, are backwards compatibility and content naming. ICN network parts with NC should be composable with ICN network parts without NC.

Names of packets and blocks are important, because they could be used for the comparison in the PIT, Content Storage and in interest packets. The system can either follow a naming scheme of unique or non-unique names. Unique naming requires every packet to have a different name but the name includes metadata like the coding vector. Non-unique naming in contrast, allows packets to be called the same, which might make renaming on inner nodes of the network necessary, if a packet name is already used on that node. The metadata is stored in the payload of the packet. Furthermore, different possibilities to decide on the name can be considered. Either the consumer is familiar with the naming conventions and already uses the right name in the interest packet, the producer decides on coding vectors and names when getting a request, or the naming scheme can be looked up by the consumer in a manifest.

Related Work. A lot of research has been conducted about NC in ICN ([30], [31], [32]). Only a few papers cite the RFC or its draft, the reason for this could be that the RFC is very recent, having only published in August 2022. In particular, the paper was named as related work by Borgia at al. in "Reliable Data Deliver in ICN-IoT Environments" [33] and by Malik et al. in "MICN: a network coding protocol for ICN with multiple distinct interest per generation" [34].

3. Conclusion

The implementation of NC in networks leads to benefits in transmission processes. A smaller payload can be received by an analytic aggregation of packets. Lost packets can be recovered by sending additional repair packets. NWCRG shows this among other research results in their papers. Besides, they identified open research challenges. One challenge, described in RFC 8975, was the "Joint Use of Network coding and Congestion Control in SATCOM System", taken on more generally in RFC 9265. Their research ties into lots of other related work as shown for each RFC. Their documents have been cited multiple times, which indicates a relevance of the research done by NWCRG.

References

[1] T. Shang and J. Liu, *Secure Quantum Network Coding Theory*. Springer Singapore, 2020.

[2] "Coding for efficient NetWork Communications Research Group (nwcrg)," https://datatracker.ietf.org/rg/nwcrg/about/, [Online; accessed 24-September-2022].

[3] C. Fragouli and E. Soljanin, "Network coding fundamentals," *Foundations and Trends® in Networking*, vol. 2, no. 1, pp. 1–133, 2007. [Online]. Available: http://dx.doi.org/10.1561/1300000003

[4] M. Zverev, P. Garrido, F. Fernandez, J. Bilbao, O. Alay, S. Ferlin, A. Brunstrom, and R. Aguero, "Robust quic: Integrating practical coding in a low latency transport protocol," *IEEE Access*, vol. 9, p. 138225 – 138244, 2021, cited by: 0; All Open Access, Gold Open Access, Green Open Access. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85117204172&doi=10.1109%2fACCESS.2021.3118112&partnerID=40&md5=1ba7c5b53dab26711f322a854176c53a

[5] L. Thomas, E. Dubois, N. Kuhn, and E. Lochin, "Google quic performance over a public satcom access," *International Journal of Satellite Communications and Networking*, vol. 37, no. 6, pp. 601–611, 2019. [Online]. Available: https://doi-org.eaccess.ub.tum.de/10.1002/sat.1301

[6] H. Wu, S. Ferlin, G. Caso, O. Alay, and A. Brunstrom, "A survey on multipath transport protocols towards 5g access traffic steering, switching and splitting," *IEEE Access*, vol. 9, p. 164417 – 164439, 2021, cited by: 6; All Open Access, Gold Open Access, Green Open Access. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85121352474&doi=10.1109%2fACCESS.2021.3134261&partnerID=40&md5=c29f81b848edb8cbc051c3f8b2fc1208

[7] Toksoy, Aral and Stubbe, Henning and Holzinger, Kilian, "NWCRG Closing Report," in *Proceedings of the Seminar Innovative Internet Technologies and Mobile Communications (IITM)*, 2022.

[8] "Coding for efficient NetWork Communications Research Group (nwcrg)," https://datatracker.ietf.org/rg/nwcrg/documents/, [Online; accessed 24-September-2022].

[9] B. Adamson, C. Adjih, J. Bilbao, V. Firoiu, F. Fitzek, S. Ghanem, E. Lochin, A. Masucci, M. Pedersen, G. Peralta, P. Saxena, and S. Sivakumar, "Taxonomy of coding techniques for efficient network communications," *RFC*, vol. 8406, 6 2018.

[10] T. Paila, R. Walsh, M. Luby, V. Roca, and R. Lehtonen, "Flute - file delivery over unidirectional transport," *RFC*, vol. 6726, 11 2012.

[11] M. Watson, A. Begen, and V. Roca, "Forward error correction (fec) framework," 10 2011.

[12] M. Watson, M. Luby, and L. Vicisano, "Forward error correction (fec) building block," *RFC*, vol. 5052, 8 2007.

[13] B. Adamson, C. Bormann, M. Handley, and J. Macker, "Nack-oriented reliable multicast (norm) transport protocol," *RFC*, vol. 5740, 11 2009.

[14] M. Luby, M. Watson, and L. Vicisano, "Asynchronous layered coding (alc) protocol instantiation," *RFC*, vol. 5775, 4 2010.

[15] M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, and J. Crowcroft, "Forward error correction (fec) building block," *RFC*, vol. 3452, 12 2002.

[16] V. Roca and B. Teibi, "Sliding window random linear code (rlc) forward erasure correction (fec) schemes for fecframe," *RFC*, vol. 8681, 1 2020.

[17] S. A. M. Ghanem, A. E. Gharsellaoui, D. Tarchi, and A. Vanelli-Coralli, "Network coding channel virtualization schemes for satellite multicast communications," *CoRR*, vol. abs/1704.04790, 2017. [Online]. Available: http://arxiv.org/abs/1704.04790

[18] K. Matsuzono, H. Asaeda, and C. Westphal, "Network coding for content-centric networking / named data networking: Considerations and challenges," *RFC*, vol. 9273, 8 2022.

[19] N. Kuhn, E. Ed, and E. Lochin, "Network coding for satellite systems," *RFC*, vol. 8975, 1 2021.

[20] S. P. Romano, C. Roseti, and A. Tulino, "Shine: Secure hybrid in network caching environment," 06 2018, pp. 1–6.

[21] B. Adamson, C. Bormann, M. Handley, and J. Macker, "Nack-oriented reliable multicast (norm) transport protocol," *RFC*, vol. 5740, 11 2009.

[22] "Multi-link routing scheme in hybrid access network with heterogeneous links," *ETSI TR*, vol. 103, pp. 351–V351, 7 2017.

[23] J. Cloud and M. Médard, "Network coding over SATCOM: lessons learned," *CoRR*, vol. abs/1506.06154, 2015. [Online]. Available: http://arxiv.org/abs/1506.06154

[24] F. Chiti, R. Fantacci, and L. Pierucci, "Energy efficient communications for reliable iot multicast 5g/satellite services," *Future Internet*, vol. 11, no. 8, 2019. [Online]. Available: https://www.mdpi.com/1999-5903/11/8/164

[25] N. Kuhn, E. Lochin, F. Michel, and M. Welzl, "Forward erasure correction (fec) coding and congestion control in transport," *RFC*, vol. 9265, 7 2022.

[26] T. Pauly, E. Kinnear, and D. Schinazi, "An unreliable datagram extension to quic," 3 2022.

[27] T. Tsugawa, N. Fujita, T. Hama, H. Shimonishi, and T. Murase, "Tcp-afec: An adaptive fec code control for end-to-end bandwidth guarantee," in *Packet Video 2007*, 2007, pp. 294–301.

[28] F. TESHIMA, H. OBATA, R. HAMAMOTO, and K. ISHIDA, "Tcp-tfec: Tcp congestion control based on redundancy setting method for fec over wireless lan," *IEICE Transactions on Information and Systems*, vol. E100.D, no. 12, pp. 2818–2827, 2017.

[29] V. Sharma, K. Kar, K. Ramakrishnan, and S. Kalyanaraman, "A transport protocol to exploit multipath diversity in wireless networks," *IEEE/ACM Transactions on Networking*, vol. 20, no. 4, p. 1024 – 1039, 2012, cited by: 45. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84865319305&doi=10.1109%2fTNET.2011.2181979&partnerID=40&md5=6bce9cc31d720be9836c2fc7bb065b41

[30] J. Zhang, L. Ma, Z. Cai, and X. Wang, "Network coding-based multipath content transmission mechanism in content centric networking," in *2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN)*, 2017, pp. 1012–1015.

[31] R. Boussaha, Y. Challal, and A. Bouabdallah, "Single-path network coding authentication for software-defined named data networking," in *2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA)*, 2019, p. 17.

[32] ——, "Authenticated network coding for software-defined named data networking," vol. 2018-May, 2018, Conference paper, p. 1115 – 1122, cited by: 3. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85052294658&doi=10.1109%2fAINA.2018.00160&partnerID=40&md5=8e21bdabfeaa8d50e9d0d89dd60587bb

[33] E. Borgia, R. Bruno, and A. Passarella, "Reliable data delivery in icn-iot environments," *Future Generation Computer Systems*, vol. 134, p. 271 – 286, 2022, cited by: 0; All Open Access, Green Open Access. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85129691482&doi=10.1016%2fj.future.2022.04.004&partnerID=40&md5=fad6f5c4b8e6289bf10921f5fdf0316b

[34] H. Malik, C. Adjih, C. Weidmann, and M. Kieffer, "Micn: A network coding protocol for icn with multiple distinct interests per generation," *Computer Networks*, vol. 187, 2021, cited by: 0; All Open Access, Bronze Open Access, Green Open Access. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85099513258&doi=10.1016%2fj.comnet.2021.107816&partnerID=40&md5=886e64b2f73b5c7a1f36f050fe411d60

Lost in Simulation: Route Property in Mininet

Philipp Kern, Henning Stubbe*, Kilian Holzinger*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: philipp.kern@tum.de, stubbe@net.in.tum.de, holzinger@net.in.tum.de

Abstract—**The Mininet network emulator enables the comparison of speed, delay, jitter and packet loss across different topologies. It provides a Python API to instantiate almost arbitrary layouts of networks and connections with attributes like predetermined packet loss. We inspect linear and grid-like topologies and discover that both share similar performance characteristics.**

Setting up more paths from one host to another does not improve latency noticeably.

Emulating more consecutive switches in a network decreases it's throughput which would not be expected in a hardware implementation.

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

1. Introduction

Many fields require dependable behaviour in their network infrastructure. To analyze how different setups influence important properties, a network emulator like Mininet [1] can be used. Mininet is a program to test and deploy network configurations and therefore provides Software-defined Networking. While it is not possible to learn how real world hardware devices react in all situations this way, the most significant attributes can be mirrored in an emulated network through placement of nodes or link properties. We want to describe important properties of network routes and measure the impact different network configurations have on them.

Using Mininet A running Mininet instance can be controlled using a command line interface. It can be configured by command line arguments, but for our investigation we are using the Python API. Shell commands can be executed from the viewpoint of each node in the network. In this research we are using iperf3 and ping to garner information on interesting properties of specific topologies. Mininet can also be instructed to drop some packets to test the resilience of TCP/IP in overloaded network connections.

2. Related Work

In [2], Torres-Jr and Ribeiro researched how packet reordering influences TCP throughput and they established *Mean Displacement* and *Entropy* metrics as simple, universally applicable basis on which to compare network behaviour.

3. Properties of Routes commonly considered by the community

Certain internet connectivity properties are considered advantageous and in some cases critical for a variety of fields.

Mininet's Limitations The properties of hardware components and physical cables are difficult to predict before they are used, which makes them hard to account for in a simulation. On the other hand, we can - with little effort - introduce certain factors like latency, bandwidth, results of having multiple paths to choose from and consequences of overloading like packet loss in Mininet.

Delay Delay is also known as lag, ping rate and latency according to the IR Team [3], who also wrote [4] about jitter and [5] about packet loss. It is important to keep delay to a minimum for playing real time online video games as movement is often precisely timed. And video conferencing with low latency enables a more natural and spontaneous conversation of all participants.

Bandwidth Network bandwidth, commonly referred to as speed, is measured in $\mathrm{bit\,s^{-1}}$. Operating a file server benefits from high bandwidth in order to provide service to many users in a given time. It also enables users to stream higher quality video from streaming platforms.

Speed Ramp-up and Consistency Speed ramp-up is likely less important in many areas than bandwidth and latency consistency, but becomes crucial in serving small web pages quickly. Websites smaller than 5 MB could not benefit from a high bandwidth connection that reaches its peak after transmitting 10 MB and being significantly slower before that. Live-streaming services additionally need consistency in bandwidth to serve their users a certain quality of video without needing to change quality often. This behaviour depends mostly on the sender's TCP implementation, namely the pace at which it enlarges its congestion window at slow start as explained in [6] by Carle et al.

Packet Loss Packet loss occurs when network hardware receives more packets than it can handle, which it will then discard requiring the sender to resend them [5]. In online speech communication it is important to keep the need for re-sending packets to a minimum in order to avoid delays. Time sensitive fields are especially dependant on low packet loss rate.

Packet Reordering To speed up traffic flow through parallelization related internet packets are sometimes split up and later arrive in the wrong order at the client. Reordering of packets can have similar consequences for video and audio calls over the internet as packet loss: voice recordings not arriving in time makes for a choppy listening experience. Additionally it strains the receiving hardware to some extent. Ghasemirahni et al. saw an increase in web server performance by manually putting packets back in order in [7].

Jitter Jitter is the variance in ping time. If that variance is too high, voice over IP communications might sound stuttered or drop out [4].

4. Deploying Topologies in Mininet

In this section we provide a basic introduction on how to deploy networks with Mininet. Command line arguments make the deployment of basic topologies easier when compared to Python scripts. A small linear topology can be created like this `sudo mn --topo=linear,4`. It contains four hosts, each being connected to one switch which are in turn connected linearly.

For the measurements conducted in this research however just using the command line is too limiting. Using the Python API directly gives control over what program is run on the nodes, how exactly they are connected and which properties they have. This gives the additional advantage of being able to parameterize the topologies more easily and automate testing. The topologies are deployed entirely within one Python script available at [8]. Three different topologies were implemented and two of them tested, all containing two hosts:

- A linear one that places a certain number of switches between the hosts which differs from the preset variant that has hosts at every switch on the line
- One with switches connected in a grid-like fashion
- A circle of switches, two of which connect to the hosts

It is possible to set the length of the linear and circular topology and the width and height of the grid-like one.

If hosts are to be used as routers in Mininet, they have to be manually configured. That is, they have to be instructed to pass on IP traffic and use specific IP addresses. However, hosts do not act as routers so connections cannot be established across them, even if they are set to forward IP traffic (see 'hostline' and 'hostangle' topologies in [8]).

5. Emulating Environments for Testing Properties

To evaluate how different network configurations affect the requirements listed in 3, we now introduce custom topologies in Mininet. The topology and test code can be found in [8]. By launching a Wireshark instance from one of the switches or hosts it is possible to inspect packages going through the respective node. To find out the achievable bandwidth and speed ramp-up we use iperf3. The goal of this is to test how the availability of many different paths affect the speed and continuity of packet flows. As such, the circular topology mentioned in 4 is too similar to the other ones and evaluating it would not yield meaningful results.

5.1. Linear Topology

To identify the impact of having to simulate a large amount of switches, we construct a topology consisting of a variable amount of switches connected in a line (Figure 1). At both ends of this line we attach a host expecting them to be able to communicate with each other. We scale this network up by adding more switches into the line, parameterizing the network by the length of this connection. When measuring and comparing to the other topology, we refer to the amount of links from the first to the second host as the path length.

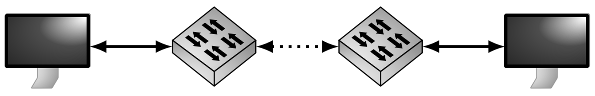


Figure 1: Linear Topology

5.2. Square Topology

Switches are placed in a two-dimensional grid and each is connected with the ones left, above, right and below itself (Figure 2). This is helpful for finding out whether alternative paths through switches can have an impact on relevant properties and how having to emulate many network devices strains the emulating system. In order for this to work at all, the Spanning Tree Protocol has to be enabled on all switches. The top left and the bottom right switch are each additionally connected to one of the two hosts. Here we define the minimum path length as the lowest number of links data, e. g. an Ethernet frame has to pass through to reach its destination which is the other host.

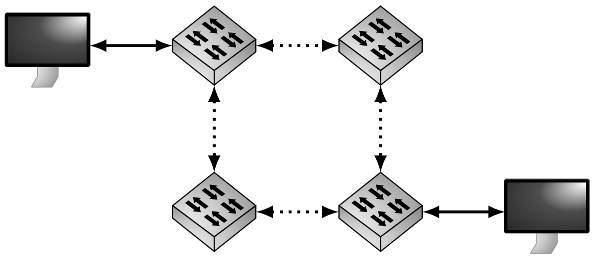


Figure 2: Square shaped Topology

5.3. Tools Used

For measuring delay, jitter and packet loss the `ping` [9] tool is used. The first ping request is sent separately from another ten as to not disturb the average times recorded from those. Bandwidth and speed ramp-up measurements are done with `iperf3` [10].

6. Impact of Parameters on Route Properties

The following tests were conducted on an Intel® Core™ i5-2520M 2.50GHz processor using the Linux Kernel version 5.19.8. The test computer has 8 GB of DDR3 RAM clocked at 1333 MHz and was running a graphical desktop environment and the VSCode IDE at the time of measurement. Although existent, the additional system load was not influential to the results as it consists mostly of background services and amounting to only about four percent CPU utilization.

6.1. Delay

To mimic high distances between communicating parties which results in higher latency we can insert a variable amount of nodes in between the two. It was found that the the additional switches did not help or hinder the ping times.

The ping command is suitable here: it displays the time it takes to send packets back and forth between clients. In general, the larger the network the more time it needs to pass on the ICMP ECHO_REQUEST datagram sent by the ping command. To compensate for this behaviour a long pause is interjected in between a first ping command, the output of which is discarded, and the consecutive *real* ones. As well as multiple routes or paths to send packets between clients, round trip times and achievable bandwidth were measured.

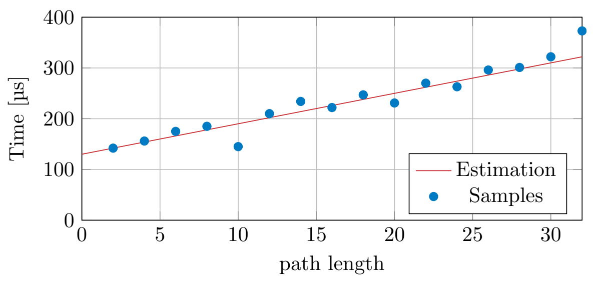


Figure 3: Average ping time in a linear topology

It appears that round trip times scale linearly with the amount of switches between the ping participants in Figure 3.

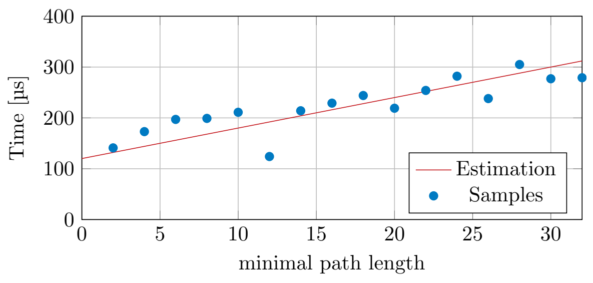


Figure 4: Average ping time in a square topology

When using a square topology with same height and width we can analyze the round trip times as well. Note that the minimal amount of links the packet has to traverse is double the amount of switches per dimension in this case. It can be observed that the additional switches in the network have not contributed to longer ping times. In fact the square shaped network turned out to be slightly faster than the linear one by on average 0.011 63 ms. In rough terms the formulas

$$d_l(x) = 0.13 + 0.006 \cdot x\text{ms} \quad (1)$$
$$d_s(x) = 0.12 + 0.006 \cdot x\text{ms} \quad (2)$$

present themselves as an estimate of the round trip time of linear and square topologies on the researcher's computer, where x is the amount of links between two hosts.

6.2. Speed Ramp-up and Consistency

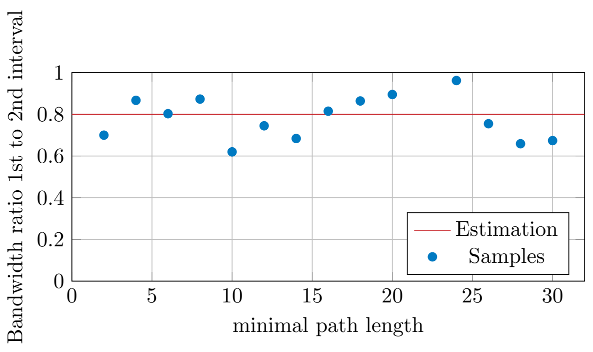


Figure 5: Speed of first 0.1 s interval relative to second interval

This test was performed in square shaped grid-of-switches-topology. Apart from two exceptions at minimum path lengths of 22 and 32, the bandwidth achieved in the first 0.1 s was always in the realm of 60 % to 100 % of the bandwidth of the second such interval (Figure 5). As a result, no significant delay would be noticeable by the user.

6.3. Packet Loss

Mininet has a setting for links to drop zero through 100 percent of packets going through. In the test scenario

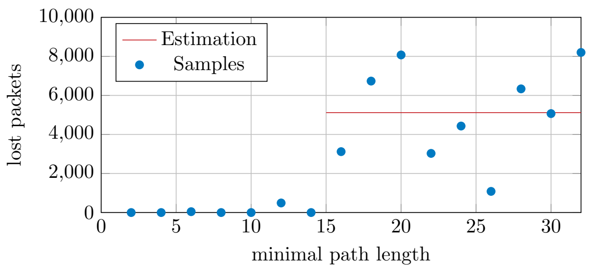


Figure 6: Packet loss in a square topology

all links were set to drop 0% of packets, but packet loss still occurred (Figure 6). When having to emulate a minimal path length of 16 or more links, on average 5119 packets were lost. This is possibly due to the emulating system not being able to handle more than a certain amount of switches. The irregularity of the specific amounts of lost packets also points toward that explanation, as system load by other applications is not uniform over time. With a correctly installed network, this should not happen in real hardware networks. The linear topology never exceeded the critical number of nodes, in this test between 52 and 66, and thus only seldomly lost a few hundred packets in two test rounds. Again other processes are to blame here.

6.4. Packet Reordering

Introducing multiple route options (e. g. hosts connected as a grid) could have led to reordered packets, but hosts were unable to connect to ones only reachable via other hosts in these tests.

6.5. Jitter

The first ping response takes longer to arrive, potentially due to having to initialize an IP connection between the hosts. Therefore, we begin our analysis of jitter at the second request, letting `ping` calculate the standard deviation of ten requests.

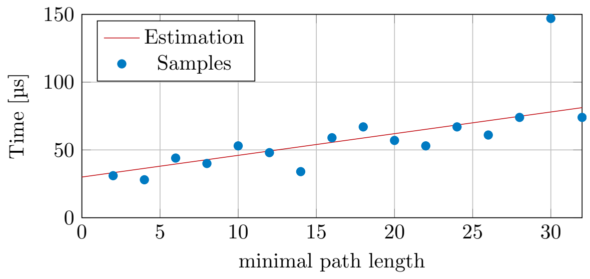


Figure 7: Standard deviation of ping in a linear topology

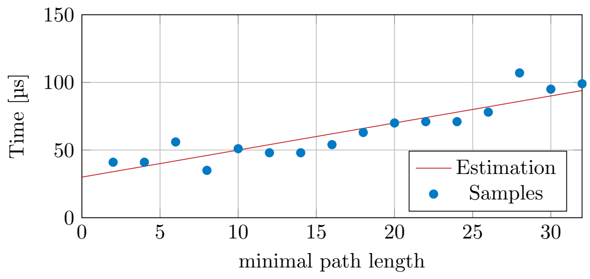


Figure 8: Standard deviation of ping in a square topology

Figures 7 and 8 indicate a linear increase in jitter with a growing number of switches with the linear topology presenting slightly more consistent round trip times.

6.6. Bandwidth

Similarly to the delay measurements, exchanging switches for hosts could have revealed changes in speed that result from having to route IP traffic in contrast to just passing on Ethernet frames. As the latter could happen without any intervention and therefore even delay in the simplest real-world case. The results of the iperf benchmarking hints toward an inversely proportional relation between path length and achievable bandwidth. The speed losses at higher node counts is best attributed to the overhead of simulation and should not have real world equivalences as switches operate independently.

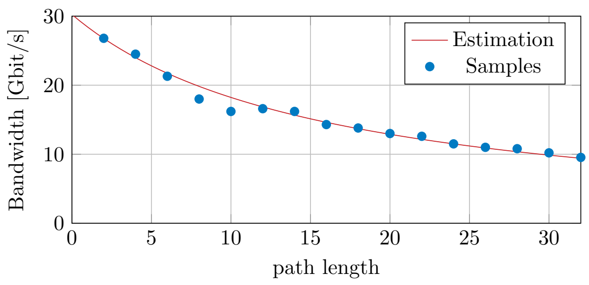


Figure 9: Average bandwidth in a linear topology

It seems that, looking at Figures 9 and 10, the availability of different paths consisting of switches from one host to another again has no great impact on speed. That is to expected, considering the simulated switches are likely incapable of rerouting traffic to avoid congestion - they are not routers after all.

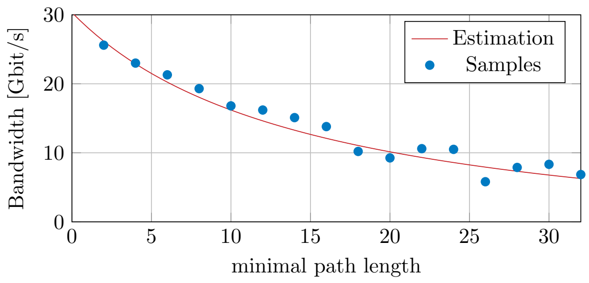


Figure 10: Average bandwidth in a square topology

The slightly better performance of the linear topology in contrast to the square one is best explained by the lower CPU load of fewer switches to simulate.

$$b_l(x) = 500 \cdot (x + 16)^{-1} - 1\text{Gbit s}^{-1} \quad (3)$$
$$b_s(x) = 530 \cdot (x + 15)^{-1} - 5\text{Gbit s}^{-1} \quad (4)$$

Formulas (3) and (4) estimate the expectable speed in Gbit s^{-1} for linear and square, topologies. However, these are not the only possibilities to predict the bandwidth, as e. g. $b_{s2}(x) = 42 - (10 \cdot \ln(x + 3))\text{Gbit s}^{-1}$ also matches the datapoints in Figure 10 closely.

7. Conclusion and Future Work

In summary, Mininet is a tool powerful in configuration options and yet simple to spin up for quick tests. Some difficulties arose from a sparsely annotated documentation of its Python API. On the other hand, by "running real kernel, switch and application code" [1], it offers the same tools as the host machine on all virtual nodes, which proved immensely helpful.

In this research we

- summarized important network properties
- presented a methodology to scale up networks for testing some of those properties
- benchmarked linear and square-shaped topologies with increasing size.

It turns out that the the ping and jitter properties of Mininet networks are reflective of hardware based networks, but that simulations sometimes lead to unrealistic

behavior. Throughput slowdown and sudden increase in packet loss are two examples for this which we found out in our research.
To find out more about certain behaviour of a networking infrastructure, Mininet supports the use of (external) controllers for topologies deployed inside it. In fixed-size topologies it would be feasible to make hosts act as routers by setting a default gateway for them manually. That allows for complex behaviour analysis of packet reordering and bandwidth and ping-time implications.

References

[1] M. P. Contributors. (2022) Mininet: An Instant Virtual Network on Your Laptop (or Other PC) - Mininet. https://mininet.org/. [Online; accessed 5-September-2022].

[2] P. R. Torres-Jr and E. P. Ribeiro, "Packet Reordering Metrics to Enable Performance Comparison in IP-Networks," *Journal of Computer Networks and Communications*, vol. 2020, p. 8, 2020.

[3] I. Team. Network Latency - Common Causes and Best Solutions. https://www.ir.com/guides/what-is-network-latency. [Online; accessed 23-September-2022].

[4] ——. Network Jitter - Common Causes and Best Solutions. https://www.ir.com/guides/what-is-network-jitter. [Online; accessed 23-September-2022].

[5] ——. What Is Network Packet Loss? https://www.ir.com/guides/what-is-network-packet-loss. [Online; accessed 23-September-2022].

[6] G. Carle, S. Günther, M. Simon, and M. Sosnowski, "Grundlagen Rechnernetze und Verteilte Systeme (GRNVS)," 2022.

[7] H. Ghasemirahni, T. Barbette, G. P. Katsikas, A. Farshin, A. Roozbeh, M. Girondi, M. Chiesa, G. Q. M. Jr., and D. Kostić, "Packet Order Matters! Improving Application Performance by Deliberately Delaying Packets," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. Renton, WA: USENIX Association, Apr. 2022, pp. 807–827. [Online]. Available: https://www.usenix.org/conference/nsdi22/presentation/ghasemirahni

[8] P. Kern. (2022) Lost in Simulation. https://gitlab.lrz.de/00000000014AB2A9/lost-in-simulation. [Online; accessed 25-September-2022].

[9] M. Muuss. (1983) ping. https://github.com/iputils/iputils. [Online; accessed 24-September-2022].

[10] J. Dugan, S. Elliott, B. A. Mah, J. Poskanzer, and K. Prabhu. (2014) iperf3. https://software.es.net/iperf/. [Online; accessed 24-September-2022].

Recent Advancements in Privacy Preserving Network Layer Approaches

Andreas Kramer, Filip Rezabek*, Richard von Seck*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: andreas.kramer@tum.de, frezabek@net.in.tum.de, seck@net.in.tum.de,

***Abstract*—Due to the rapid growth of internet communications, privacy becomes ever more important in our digital age. But since cryptography is not enough to preserve the users' privacy, solutions on the network layer become crucial. Tor and I2P offer their users privacy protection with considerable performance but have some robustness drawbacks. Therefore, we introduce the recently released anonymity infrastructure Nym that protects privacy on the network layer and addresses certain technical shortcomings of the currently most well-known anonymity networks. In this survey, we find out that for Tor and I2P a wide range of vulnerabilities of privacy against global adversaries are known, while Nym offers a high level of privacy even against this kind of attacker. However, privacy is not always the only goal that such solutions must achieve because the quality of service and an acceptable level of latency must also be maintained. A future challenge for Nym is to identify whether incentivizing its operators leads to a larger user base, which would result in the required and desired performance as well as increase the strength of the anonymity properties.**

***Index Terms*—privacy, quality of service, mix nets, onion routing, i2p, nym**

1. Introduction

In the 1990s the U.S. government tried to constrain the use of strong cryptography to ensure that national security and law enforcement agencies could break all ongoing encrypted communications via backdoors. This led to a heated debate known as the "Cryptowars" [1]. In 1993, Eric Hughes wrote the Cyphernomicon, a pamphlet arguing for cryptography with the fundamental goal of achieving and supporting personal privacy in the digital world:

> "Privacy is necessary for an open society in the electronic age. Privacy is not secrecy. A private matter is something one does not want the whole world to know, but a secret matter is something one does not want anybody to know. Privacy is the power to selectively reveal oneself to the world." [2]

But protecting only the content of the user messages does not automatically preserve its privacy. Due to the structure of the Internet protocol (IP), the OSI layer three plays a crucial role when privacy should be preserved. In addition to the payload, an IP datagram consists of a header that contains metadata such as the source and destination address [3]. The application of cryptography helps to protect the confidentiality and integrity [4] of messages, but even if all possible fields are encrypted and none of this header information is revealed, attackers can still detect communication patterns. Therefore, the difficulty of eavesdropping packets in the network and traffic analysis, including matching the amount of data or examining connection establishment or termination [5], should be increased to defend the users' privacy. But privacy comes with the cost of latency. Thus, often a trade-off between privacy and Quality Of Service (QoS) exists. Bounded Privacy is describing this problem where for a given threshold on the QoS a feasible level of privacy is guaranteed [6]. This survey focuses on solutions that aim to protect the privacy of its users and hence also its meta data. In Sections 2 and 3, a definition of privacy and the privacy-enhancing technologies mix network and onion routing are given and used to argue about concrete implementations of these technologies like I2P [7] and Tor [8], which offer a good level of privacy but still have weaknesses, especially against powerful adversaries that can watch the entire network and traffic. Additionally, in Section 4 the newly released privacy infrastructure Nym [9] is described, which is based on a mix net system and economic incentives for operating components of it and aims to solve these problems.
In Section 5, the presented solutions are then compared regarding their weaknesses against a global adversary, a conclusion is drawn, as well as future work is named.

2. Background

In this section we introduce a definition of privacy and the attacker models as which later described solutions can be attacked.

2.1. Attacker Models

To precisely describe the attacks, we define and distinguish different attacker models by their capabilities. An attacker may control various subsets of nodes in a network. Depending on the distribution of the nodes, an attacker may gain more or less information.
Depending on the network position there are **external** and **internal** attackers. An **external** adversary can compromise the communication links and an **internal** one participates in the anonymous system and therefore can compromise connections or peers. These are of special interest as they provide routing or enhanced security functions [10].
Depending on the geographic location we distinguish between **global** and **local** attackers. An adversary with

access to all communication links is called **global** while a **local** one can just act on specific connections or peers. Attackers that can just eavesdrop on the communication medium are defined as **passive**. **Active** ones can delay, modify, and omit messages and may be able to compromise peers.
The standard threat model for symbolic protocol analysis is the **Dolev-Yao model (DY)** [11]. This adversary can read, modify, destroy all message traffic, and perform any operation possible for a normal protocol user without breaking the cryptographic primitives [12]. In literature, a protocol that is secure under DY is also seen as secure under a less powerful attacker [13], [14].

2.2. Privacy

According to A. Pfitzmann and M. Köhntopp [15] privacy is defined by the terms anonymity, pseudonymity, unlinkability, and unobservability. These provide protection against the discovery and misuse of identity by other users [16]. In order to show the properties of privacy-enhancing technologies we now introduce the mentioned terms.
Anonymity is defined as not being identifiable from other subjects in the same set which is called *anonymity set*. We unite all subjects that might cause an action in that anonymity set. Depending on that set, there exist two different types of anonymity. The *sender anonymity set* is the subset of all subjects that might send traffic in the same network. The *recipient anonymity set* is the subset of all subjects that might receive traffic in the same network. These sets may be disjoint but also overlap. In general, one can state that anonymity becomes stronger with the size of the set [15].
Pseudonyms, like identifiers, are used to not reveal data about subjects during specific actions. Nevertheless, user actions can still be linked with pseudonyms by the system itself. *Pseudonymity* ensures that users may use resources or services without revealing their identity so that they are still accountable for their use [16].
Unlinkability is described as the inability to connect or combine subjects with initially separate information. This means that the probability of finding a relation between those items stays the same before (a-priori knowledge) and after an action within a system (a-posteriori knowledge of the attacker) [15].
Unobservability requires that subjects cannot determine whether a specific action was performed. This means that the Items Of Interest (IOI) are indistinguishable from other IOIs. [15].

3. Existing Privacy-Enhancing Solutions

In this section, we take a closer look at solutions that offer network privacy. The anonymity concepts mix network and onion routing are described as a basis to argue about concrete implementations, their characteristics, and their weaknesses.

3.1. Anonymity Concepts

A **mix network (mix net)** is an overlay network of mix nodes that routes messages through the network anonymously [17].

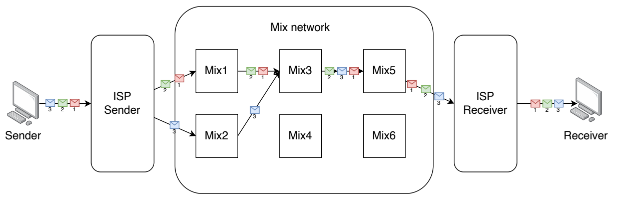


Figure 1: Mix network infrastructure

Secure mix nets can be classified as Decryption-Mix Network (DMN) which was initially proposed by Chaum [18] already in 1981, and Re-Encryption Mix Network (RMN), a concept by Park et al. [19]. A mix net protocol run includes a set of senders S_1, ... , S_n, the mix servers M_1, ... , M_n and a public bulletin board B. RMNs additionally have trustees T_1, ... , T_n. The senders transmit their ciphertexts to the mix servers which add delay and then publish them in random order. Channels are used to ensure that eligible senders S_i securely submit messages to the bulletin board B. The protocol run is split into three phases. In the *setup phase*, all required parameters are generated. The *submission phase* is then used to generate and submit the senders' messages. Lastly, in the *mixing phase*, the mix servers collaboratively mix the input.
The purpose of a mix net is to provide unlinkability, so to hide the links between the communication partners and their messages [18]. This can be assured by delaying the messages and then shuffling (also called mixing) before forwarding them. Figure 1 displays both the network position of such mix nets and illustrates the message shuffling. The sent messages are fixed-sized due to message padding, where random data is attached to messages of deviating size [4]. Additionally, whenever the user does not have any actual payload to send to the mix network, the client sends instead loop cover packets, which are messages with dummy payload that have the same receiver as sender. This leads to the indistinguishability of real messages from cover messages and therefore to unobservability [20]. Mix nets are hence designed to provide meta data protection against global adversaries. As long as not all mix servers of the message path are corrupted, the mix net can guarantee sender anonymity.
With *Decryption-Mix Networks* the sender is required to iteratively encrypt the input messages m_i with the public keys pk_1, ... , pk_n of the mix servers M_1, ... , M_n. This can be achieved with public key cryptographic systems like RSA. The message is encrypted in reverse order and the resulting layered ciphertext c_i is then submitted to the first mix server M_1. The mix server M_i then uses his private key sk_i to decrypt the outermost encryption layer of all input ciphertexts, shuffles the decrypted messages and forwards them to the next mix server M_{i+1}. The last mix server could then output the plain messages initially chosen by the senders in random order. To offer a higher level of privacy messages are stored until an adequate threshold is reached and they are then forwarded to the next hop [20].
For *re-encryption mixnets* we need to use a public-key encryption scheme that allows for re-encrypting a given ciphertext without knowing the secret key or the encrypted message like ElGamal, an asymmetric key encryption algorithm for public-key cryptography [21].

As mentioned before RMNs additionally use trustees t_1, ..., t_n with whom each sender s_i shares the secret key of its public key pk. With this information, the sender s_i encrypts its message. The mix server M_i then re-encrypts all ciphertexts with coins chosen independently and uniformly at random, shuffles the re-encrypted ciphertexts, and forwards them to the next mix server M_{i+1}. This procedure is repeated until the last mix server is reached. It then outputs a list of ciphertexts that encrypt the input messages initially chosen by the senders but under different random coins and in a random order [20].

Onion Routing is a general-purpose infrastructure for privacy in public networks that allows lower latency than mix nets as messages are not mixed or delayed [22].

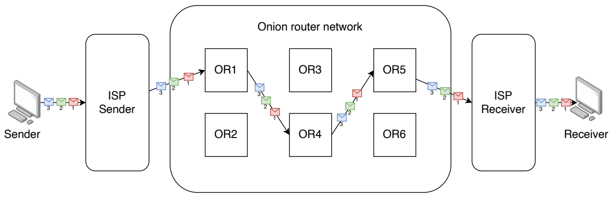


Figure 2: Onion routing network infrastructure

Similar to mix nets, messages are sent over multiple Onion Routers (ORs) inside the network. As shown in Figure 2, the messages are relayed through the OR network without delaying them and thus without shuffling. This allows in comparison to mix nets a lower latency. ORs work as intermediate proxies, their physical location in the network is unknown [23]. The onion data structure sent to the ORs consists of several encryption layers around the payload. ORs accept fixed-length messages through message padding. Additionally, they perform cryptographic operations like removing one layer of encryption using its own private key on the messages like a DMN before forwarding it to the next mix server. The path through the network is defined by the client, which builds a circuit. Every OR knows only its predecessor and the successor but has no further information about other routers in its circuits like the origin, destination, or the payload. As described before, in DMNs the mix servers introduce a delay before forwarding their messages. Onion routing works without these delays, which may lower the anonymity [8], [22] but also lower the latency.

3.2. Anonymity Networks

Tor is a distributed overlay network that offers anonymity on TCP-based communications in networks and is based on the onion routing protocol described in Section 3.1 [8]. It additionally includes several improvements like perfect forward secrecy [24], hidden services, which provide receiver anonymity, and rendezvous points to connect to them [8]. Tor is not fully distributed as it uses directory services to store statistics and information about Tor nodes. It cannot defend against end-to-end correlation attacks from global adversaries because it does not employ delays for cells. Also, deanonymizing of hidden services is possible [8], [25], [26]. But even local attackers can determine the visited webpages with fingerprinting attacks [27]. As mentioned before, the anonymity of a system depends also on the size of the anonymity set. This means for Tor more hops (ORs) and more users may lead to higher anonymity. As the Tor network does not incentivize the operation of components like routers, the network size remained roughly the same in the last years [28].

The **Invisible Internet Project (I2P)** is a fully encrypted private network layer on the basis of a peer-to-peer network. I2P, like Tor, uses a variant of onion routing named garlic routing to create anonymous connections [7].

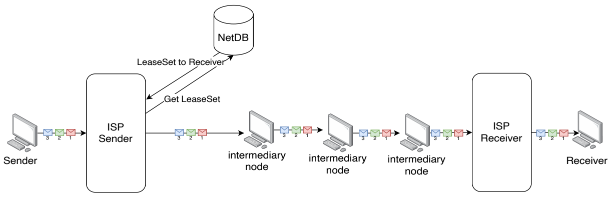


Figure 3: Architecture of a I2P network

The participants work both as clients and as proxy routers that forward the messages sent through the network. To achieve a fully distributed architecture, I2P uses a Network Database (NetDB), which is implemented as a Distributed Hash Table (DHT) using the Kademlia algorithm [29]. In NetDB information about peers and services is saved. To communicate with other peers the sender has to get leaseSets from the database, which contain data such as public keys for communication [30]. Figure 3 depicts the message flow through the I2P network and the preliminary request to NetDB. Nevertheless, I2P has some weaknesses. Its anonymity set is small because of the modest size of the current network [31]. A consequence is the current bad performance of its services, because of the overhead for encryption and routing which limits the bandwidth. Also, effective defenses against Sybil attacks remain an open question [31].

3.3. Summary and Challenges of Solutions

As described in Section 3.2 the currently two most well-known anonymity networks Tor and I2P still suffer from some weaknesses, especially against global, passive, or stronger adversaries. Another problem of both remains the lack of growth that comes with latency and leads to a smaller anonymity set. With an increased number of participants, some of the weaknesses would be reduced [31], [32]. Tor got a lot of attention from researchers which led to constant improvement and good documentation. I2P on the other hand got less attention due to the missing clear design documentation [17]. Nevertheless, in both networks, the key components are run on a volunteer basis which led to a consistent number of operators. Nym now explores the impact of node incentives. This mechanism should not only lead to more node operators in the network but also prevent freeloading [33] and limit the possibilities of malicious users in the network [17].

4. Nym

This section describes the newly released anonymity infrastructure Nym. Its design goal is to support privacy-enhanced access to applications and services. Node operators are incentivized by their own tokens, named Nym tokens, to support the operational costs with proof of mixing. Proof of mixing is defined as incentivizing node operators to correctly and reliably process, route, and deliver messages. This shall lead to dynamic scaling for privacy and high quality of service.

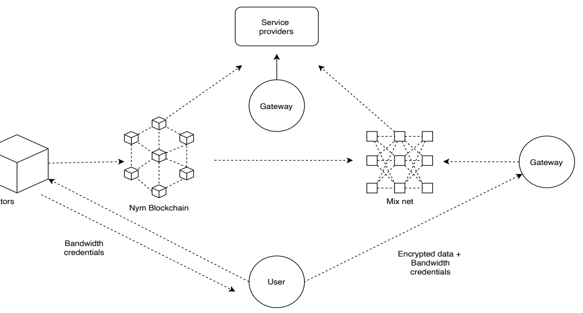


Figure 4: Nym architecture [34]

Figure 4 depicts Nym's architecture and its three types of nodes: **Validators**, **Gateways**, and **Mix Nodes**.

Mix Nodes. Mix nodes provide communication privacy and are part of the mix network. Nym uses a Loopix [35] design for its mix net, modified to provide better QoS guarantees. Loopix is a low-latency anonymous communication system that provides sender and receiver anonymity as well as unobservability. As described in Section 3.1 mix nodes receive data packets that they transform cryptographically and reorder. Sphinx [36] is used as a packet format, which is cryptographic and relays anonymized messages within the mix network. It supports indistinguishable replies, hides the path length and relay position as well as provides unlinkability. Loopix also uses a continuous-time mixing strategy, where each message is delayed individually and independently of the others [37]. The concrete mixing delay is chosen by the original sender and encoded in the Sphinx header. Additionally, Loopix applies dummy traffic and message padding. This ensures a minimum level of anonymity at all times, obfuscates the timing and volume of user communication, and therefore achieves unobservability [9], [17].

Gateways. Gateways mediate the access to the network and its services. They act as proxies between the mix net and the participants. Users may always choose the same gateway for all their traffic or multiple ones. Gateways cache received messages for offline or unreachable participants. Users need to show valid unspent bandwidth credentials, provided by the validators, to send messages through the mixnet [9], [17].

Validators. Validators maintain the Nym blockchain and handle transactions for two types of credentials. Bandwidth credentials prove the right to send traffic through the mixnet. Service credentials can encode arbitrary attributes for the proof of access rights for a service [17]. These credentials are provided with a modified version of coconut, a cryptographic signature scheme, which supports decentralized credential issuance and thus is resistant to local adversaries [9], [38]. The Nym blockchain works as a broadcast channel for network-wide information and includes data like the list of nodes and their public keys, network configuration parameters, or participants' stake.

Service providers. Nym as an infrastructure supports privacy for third-party applications and services that are accessible through the network. These can send and receive messages to privately communicate and use the Nym credentials to grant access to their services [17].

5. Comparison of Solutions

In this section, we compare the offered privacy for the presented anonymity networks against global adversaries. Table 1 depicts the known vulnerabilities ordered by in Section 2.2 defined privacy terms. The concrete value describes whether vulnerabilities for the solution on given requirement exist or not. The table is only valid for this type of attacker and the current amount of users.

	Tor	I2P	Nym
Anonymity	Yes	Yes	No
Unlinkability	Yes	Yes	No
Unobservability	Yes	Yes	No

TABLE 1: Known vulnerabilities of privacy requirements against global adversaries

Tor in general does not provide protection against global attackers [39], [40]. As described in Section 3.2, onion routing encrypts the messages between each OR so that only the last hop can see the decrypted message. But still, Tor suffers weaknesses against global adversaries, because though the packets are encrypted, Tor does not add timing obfuscation to conceal the traffic patterns. Additionally, Tor's design uses a centralized directory authority to build tunnels through the network which may be another attack vector. As shown by the different deanonymization attacks, Tor suffers known vulnerabilities in anonymity against this kind of attacker [41], [42]. Also, unlinkability cannot be preserved, as shown by attacks [43]. As no cover traffic is included in either the onion routing protocol or in Tor unobservability cannot be guaranteed either [44].
Though I2P is based on a peer-to-peer architecture, like Tor the network cannot defend against global attackers. I2P replaces Tor's directory authority with a DHT for routing. As described in Section 3.2 the usage of a distributed hash table may be an attack vector. It is open to several attacks that isolate, misdirect, or deanonymize users like brute-force, timing, or intersection attacks [7], [30], [31]. Therefore, I2P is not able to guarantee anonymity or unlinkability against global adversaries. I2P also does not use cover traffic. Due to that reason, the network cannot grant unobservability [31], [44].
Nym aims to protect also against global adversaries by the usage of the in Section 4 described Loopix mix net. Mentioned Loopix mix net aims to protect the users' unlinkability [9], [17]. Neither Loopix nor Nym have so far known vulnerabilities of anonymity, unlinkability, or unobservability. Therefore, all values in Table 1 are no. Since Nym modified Loopix to provide a better QoS it cannot be ruled out that there are yet undocumented vulnerabilities. Comparable to the vulnerabilities of Tor's directory services, the gateways, for example, could provide an attack vector. Unlike I2P or Tor, Nym adds cover traffic and timing obfuscation, which should prevent unobservability [17], [45]. Nevertheless, it is important to mention that not much research could be conducted yet to identify possible weaknesses.

6. Conclusion and Future Work

In this paper, we provided an overview of recent anonymity concepts and networks as well as a comparison of them with the newly released Nym. Section 5 has shown that, with current knowledge, Nym provides a high level of privacy for its users, even against global attackers, considering that the solution is relatively new and not much research has been done on it. However, the privacy provided by mix networks comes with latency, where always a trade-off between privacy and QoS exists. The question in the future will be whether the market will adopt the concept chosen by Nym tech of rewarding their operators, so that with a larger user base, also more operators will run mix servers, validators, or gateways, so that the required and desired performance is given in addition to privacy. Future challenges and opportunities lie in the question of how the latency currently compares to the other solutions described in this survey and how it scales with the addition of more mix servers. Low latency and thus good performance could be next to privacy a major argument to use Nym. And a large amount of users would increase the strength of anonymity properties.

Acronyms

DHT	Distributed Hash Table. 3, 4
DMN	Decryption-Mix Network. 2, 3
DY	Dolev-Yao model. 2
I2P	Invisible Internet Project. 3, 4
IOI	Items Of Interest. 2
IP	Internet protocol. 1
NetDB	Network Database. 3
OR	Onion Router. 3, 4
OSI	Open Systems Interconnection. 1
QoS	Quality Of Service. 1, 4
RMN	Re-Encryption Mix Network. 2

References

[1] B.-J. Koops and E. Kosta, "Looking for some light through the lens of "cryptowar" history: Policy options for law enforcement authorities against "going dark"," *Computer Law and Security Review*, vol. 34, no. 4, pp. 890–900, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0267364918302413

[2] E. Hughes, "A Cypherpunk's Manifesto," https://www.activism.net/cypherpunk/manifesto.html, 1993, [Online; accessed 10-October-2022].

[3] J. Postel, "Internet Protocol," RFC 791 (Internet Standard), RFC Editor, Fremont, CA, USA, Sep. 1981, updated by RFCs 1349, 2474, 6864. [Online]. Available: https://www.rfc-editor.org/rfc/rfc791.txt

[4] M. Bishop, *Introduction to Computer Security*. Addison-Wesley Professional, 2004.

[5] J. Ren and J. Wu, "Survey on anonymous communications in computer networks," *Computer Communications*, vol. 33, pp. 420–431, 03 2010.

[6] L. Hartmann, "Bounded Privacy: Formalising the Trade-Off Between Privacy and Quality of Service," in *SICHERHEIT 2018*, H. Langweg, M. Meier, B. C. Witt, and D. Reinhardt, Eds. Bonn: Gesellschaft für Informatik e.V., 2018, pp. 267–272.

[7] "Invisible Internet Project (I2P)," https://geti2p.net, [Online; accessed 16-September-2022].

[8] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation Onion Router," in *13th USENIX Security Symposium (USENIX Security 04)*. San Diego, CA: USENIX Association, Aug. 2004. [Online]. Available: https://www.usenix.org/conference/13th-usenix-security-symposium/tor-second-generation-onion-router

[9] "NYM - Building the next generation of privacy infrastructure," https://nymtech.net, [Online; accessed 16-September-2022].

[10] R. Staudemeyer, D. Umuhoza, and C. Omlin, "Attacker models, traffic analysis and privacy threats in IP networks," 01 2005, p. 7.

[11] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–208, 1983.

[12] Q. Chen, C. Zhang, and S. Zhang, *Secure Transaction Protocol Analysis: Models and Applications*, 01 2008, vol. 5111.

[13] W. Arsac, G. Bella, X. Chantry, and L. Compagna, "Multi-Attacker Protocol Validation," *J. Autom. Reasoning*, vol. 46, pp. 353–388, 04 2011.

[14] I. Cervesato, "The Dolev-Yao Intruder is the most Powerful Attacker," 2010.

[15] H. Federrath, *Designing Privacy Enhancing Technologies International Workshop on design issues in anonymity and unobservability, Berkeley, CA, USA, July 25-26, 2000. proceedings*. Springer Berlin Heidelberg, 2001, pp. 1–9.

[16] "Common Criteria for Information Technology Security Evaluation," *Part 2: Security functional components*, 2017.

[17] Diaz, Halpin, and Kiayisas, "The Nym Network - The Next Generation of Privacy Infrastructure," vol. 1.0, 2021.

[18] D. L. Chaum, "Untraceable Electronic Mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 24, no. 2, p. 84–90, 1981.

[19] C. Park, K. Itoh, and K. Kurosawa, "Efficient Anonymous Channel and All/Nothing Election Scheme," in *Workshop on the Theory and Application of Cryptographic Techniques on Advances in Cryptology*, ser. EUROCRYPT '93. Berlin, Heidelberg: Springer-Verlag, 1994, p. 248–259.

[20] T. Haines and J. Müller, "SoK: Techniques for Verifiable Mix Nets," in *2020 IEEE 33rd Computer Security Foundations Symposium (CSF)*, 2020, pp. 49–64.

[21] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985.

[22] D. Goldschlag, M. Reed, and P. Syverson, "Onion Routing for Anonymous and Private Internet Connections," *Communications of the ACM*, vol. 42, 02 1999.

[23] N. Dutta, N. Jadav, S. Tanwar, H. Deva Sarma, and E. Pricop, *Cyber Security: Issues and Current Trends*, 01 2022.

[24] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography*, 1st ed. USA: CRC Press, Inc., 1996.

[25] A. Biryukov, I. Pustogarov, and R.-P. Weinmann, "Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization," in *2013 IEEE Symposium on Security and Privacy*, 2013, pp. 80–94.

[26] R. Jansen, F. Tschorsch, A. Johnson, and B. Scheuermann, "The Sniper Attack: Anonymously Deanonymizing and Disabling the Tor Network," 01 2014.

[27] M. Juarez, S. Afroz, G. Acar, C. Diaz, and R. Greenstadt, "A Critical Evaluation of Website Fingerprinting Attacks," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 263–274. [Online]. Available: https://doi.org/10.1145/2660267.2660368

[28] "Tor Metrics," https://metrics.torproject.org/networksize.html, [Online; accessed 21-September-2022].

[29] P. Maymounkov and D. Eres, "Kademlia: A peer-to-peer information system based on the xor metric," vol. 2429, 04 2002.

[30] C. Egger, J. Schlumberger, C. Kruegel, and G. Vigna, "Practical Attacks against the I2P Network," in *Proceedings of the 16th International Symposium on Research in Attacks, Intrusions, and Defenses - Volume 8145*, ser. RAID 2013. Berlin, Heidelberg: Springer-Verlag, 2013, p. 432–451. [Online]. Available: https://doi.org/10.1007/978-3-642-41284-4_22

[31] "Invisible Internet Project (I2P): Threat models," https://geti2p.net/de/docs/how/threat-model, [Online; accessed 20-September-2022].

[32] E. Erdin, C. Zachor, and M. H. Gunes, "How to Find Hidden Users: A Survey of Attacks on Anonymity Networks," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 4, pp. 2296–2316, 2015.

[33] Z. Zhang, W. Zhou, and M. Sherr, "Bypassing Tor Exit Blocking with Exit Bridge Onion Services," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 3–16. [Online]. Available: https://doi.org/10.1145/3372297.3417245

[34] "Choose your Character: an Overview of Nym Network Actors," https://blog.nymtech.net/choose-your-character-an-overview-of-nym-network-actors-19e6a9808540, [Online; accessed 10-October-2022].

[35] A. M. Piotrowska, J. Hayes, T. Elahi, S. Meiser, and G. Danezis, "The Loopix Anonymity System," in *Proceedings of the 26th USENIX Conference on Security Symposium*, ser. SEC'17. USA: USENIX Association, 2017, p. 1199–1216.

[36] G. Danezis and I. Goldberg, "Sphinx: A Compact and Provably Secure Mix Format," in *2009 30th IEEE Symposium on Security and Privacy*, 2009, pp. 269–282.

[37] G. Danezis, "The traffic analysis of continuous-time mixes," in *Proceedings of the 4th International Conference on Privacy Enhancing Technologies*, ser. PET'04. Berlin, Heidelberg: Springer-Verlag, 2004, p. 35–50. [Online]. Available: https://doi.org/10.1007/11423409_3

[38] A. Sonnino, M. Al-Bassam, S. Bano, and G. Danezis, "Coconut: Threshold Issuance Selective Disclosure Credentials with Applications to Distributed Ledgers," 02 2018.

[39] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr, *Towards an Analysis of Onion Routing Security*, 03 2001, pp. 96–114.

[40] S. Chakravarty, A. Stavrou, and A. D. Keromytis, "Approximating a Global Passive Adversary Against Tor," 2008. [Online]. Available: https://academiccommons.columbia.edu/doi/10.7916/D82J6QBM

[41] F. Buccafurri, V. Angelis, M. Idone, C. Labrini, and S. Lazzaro, "Achieving Sender Anonymity in Tor against the Global Passive Adversary," *Applied Sciences*, vol. 12, p. 137, 12 2021.

[42] I. Karunanayake, N. Ahmed, R. Malaney, R. Islam, and S. K. Jha, "De-Anonymisation Attacks on Tor: A Survey," *IEEE Communications Surveys and Tutorials*, vol. 23, no. 4, pp. 2324–2350, 2021.

[43] S. Murdoch and G. Danezis, "Low-cost traffic analysis of Tor," in *2005 IEEE Symposium on Security and Privacy*, 2005, pp. 183–195.

[44] J. Ren and J. Wu, "Survey on anonymous communications in computer networks," *Computer Communications*, vol. 33, pp. 420–431, 03 2010.

[45] "NYM - Documention," https://nymtech.net/docs, [Online; accessed 22-September-2022].

$P4_{16}$: Out of the Loop

Felix Sandmair, Henning Stubbe, Eric Hauser*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: felix.sandmair@tum.de, stubbe@net.in.tum.de, hauser@net.in.tum.de

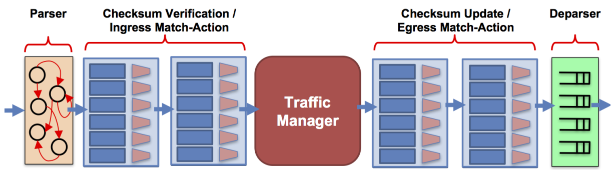
***Abstract*—$P4_{16}$ is a programming language used for packet processing that was formulated without providing loops, such as for or while loops. This Paper is going to give a quick summary of $P4_{16}$ in order to show how it is still possible to implement loops in $P4_{16}$ and afterward compare the solutions in regard to performance.**

1. Introduction

$P4_{16}$ is a programming language that differs from many programming languages because it was designed without a concept for loops. The reasoning behind that is that if we eliminate iterations, the time needed for computing a P4 program is linearly dependent on the packet size. Having no loops can be limiting at times, e. g. for processing the payload of the packet, you might need some form of looping [1]. Another example would be TCP SYN proxies. TCP SYN proxies are a measure to protect a server against SYN flooding attacks [2]. To implement this functionality on software-defined networks looping would be very helpful. Furthurmore, this investigation shows the limits of the programming language $P4_{16}$. The following sections will discuss how it is possible to write loops in $P4_{16}$ even without concepts like while and goto. After explaining the approaches, it will focus on analyzing the performance and comparing our solutions.

2. $P4_{16}$ Introduction

$P4_{16}$ is a language that is used to program the data plane of software-defined networks. The programs written in this language are deployed on programmable networking hardware like routers, switches, network interface cards, or network appliances. Those devices are called targets. The manufacturer of those targets needs to provide the hardware or software implementation framework, an architecture definition, and a P4 compiler. $P4_{16}$ can only specify the data plane functionality of a device. After compilation, it generates an API for the control plane to interact with the data plane. The architecture definitions describe how the architecture is put together. Each $P4_{16}$ program can be divided into multiple Blocks, usually always consisting of a Parser, a Deparser, and a variable amount of Control Blocks. In this paper, the V1 Architecture will be the architecture that is used in the $P4_{16}$ programs, which is an architecture by the P4 Language Consortium. This architecture was designed in a way that is comparable to the old $P4_{14}$ architecture. Figure 1 shows how the architecture of the V1 Model is structured.
The Parser is always the first block a packet passes through. The Parser resembles a finite state machine that parses the packet headers and extracts the header data into a header struct. This data could be an ethernet header, but also personalized headers not following any popular protocol. Since it is possible to specify any header structure, it is technically possible to parse more than just the headers, e. g. a packet's payload [1].


Parser
Checksum Verification / Ingress Match-Action
Traffic Manager
Checksum Update / Egress Match-Action
Deparser
Figure 1: V1 Model Architecture [3]

After the parser, there are different programmable control blocks. In the case of the V1 Architecture, there are the MyVerifyChecksum, MyIngress, MyEgress, and MyComputeChecksum control blocks. Every target may also have extra functionality it can provide the user of the target. This functionality can be provided to the programmer with a so called extern function. These extern fuctions are methods that are unique to different targets. The Architecture can also limit the programmer to use certain extern functions only in certain control blocks [3]. In the end, the packet gets reassembled by the deparser. The deparser takes all the header data that was manipulated within the control blocks and maybe added new ones and puts them back together into a network packet.

3. Loop Concepts

In this section, the different approaches to implement loops in $P4_{16}$ will be discussed.

3.1. Parser Loops

Parser Loops are probably the most straightforward possibility to program loops in $P4_{16}$. At the beginning of the parser, there will be the opportunity to declare variables and constants. Also, the objects needed inside of the loop, can be instanciated here. After that, starts the definition of the finite state machine. The finite state machine always consists of a initial state, called start, where the parsing of every packet will begin and two different final states, accept and reject. Usually, each state will be used to extract the headers of the received packets. The headers extracted will be saved in a C struct like header struct. After the extraction of the header the transition will be invoked based on the information extracted from the header. E.g. if the header extracted would be an ethernet header and the EtherType of the header would indicate that the packet is an IPv4 packet it goes to the state for extracting IPv4 header. This decision will be done by using a select statement that matches an expression list to specific values. After the select statement matches the expression list, it will transition to the given state. Before the transition statement, an arbitrary amount of parser statements can be written. The syntax of Parser statements allows basic arithmetic operations on variables and constants and methods calls on objects instantiated before. Depending on the target, the parser statements might allow more or less functionality inside the parser statements. E.g. Not all targets will allow the use of if statements inside the parser. In case if statements are needed inside the loop it is possible to use the select statement to achieve the same functionality a if statement would provide. This will come at the cost of one state per if statement used inside the loop. A significant benefit of this approach to building loops in $P4_{16}$ is that it can be done on nearly every target, but there are exceptions. Some targets will not compile parsers that contain loops. The compiler will check if the parser can be unrolled into a graph without circles at compile time. If the parser can not be unrolled, it will further check if the loop advances the cursor of the packet in every iteration. If this is the case, the parser can be compiled since the packet size is finite and if the cicle always advances the cursor the loop can not loop infinitely [4].

```
bit<32> i = 0;

    state start {
        transition l1;
    }

    state l1 {
      //loop-body
      i = i + 1; //condition-update
      transition select(i<=CYCLES/*while-condition*/) {
          true:l1;
          default:parse_ethernet;
      }
    }
```

Source Code 1: do-while-loop in parser

```
bit<32> i = 0;

    state start {
        transition l1;
    }

    state l1 {
      transition select(i<=CYCLES/*while-condition*/) {
          true:l2;
          default:parse_ethernet;
      }
    }

    state l2 {
      //loop-body
      i = i + 1; //condition-update
      transition l1;
    }
```

Source Code 2: while-loop in parser

In Source Code 1 and Source Code 2 , two different implementations of parser loops can be seen. The code fragments show an implementation of a do-while-loop and the regular while-loop. For both solutions, the declaration of all variables needed and instanciation of all objects takes place. In this case, the variable i, which is used to cycle through the loop a constant amount of times, is the only variable needed. After that, the implementations differ. For the do-while-loop in Source Code 1, the body of the loop can be executed immediately before checking any condition. Afterwards, the breaking condition for our loop is written inside the select statement. If the condition holds, the select statement will evaluate to true and transition back to the same state. In order to get the more commonly used while-loop the usage of an additionale state is obligatory. This time instead of executing the loop-body inside the first state, the condition is beeing checks first. If the condition is met, the select statement transition to the second state of the loop l2, to execute the loop-body. After the loop execution is finished, l2 transitions back to state l1. The loop body of this example only consists of one line where the variable i gets incremented to iterate a fixed amount of times through the loop. Back in state l1 the condition check happens again. When the point that the condition evaluates to false is reached, it is possible to into another loop or in this case start with parsing of the headers of the packet.
Two different solutions were beeing implemented in hope of performance benefits if only one state is beeing used instead of two different states because of less transitioning between states. Another reason is to show that there are more then just one possibility to introduce loops into the parser.

3.2. Recirculate Loops

Another possibility to realize loops is by using extern functionality. In this case, the extern method recirculate_preserving_field_list() provided by the V1 architecture was beeing used. This extern simply takes the packet that is currently processed deparses it and introduces it back into the packet processing beginning with the parser. This method can only be used in the MyEgress control block of a P4 Program. When using this method, it receives a number as a parameter. This number represents a list of variables from the metadata that should be saved for the next pass through the packet processing. This means every variable that is needed for the loop, like a loop index, is stored inside the metadata and annotated using the @field_list annotation. The number used to annotate the varibales is given to the recirculate_preserving_field_list() extern as a parameter.

```
struct metadata {
    @field_list(RECIRC_FL)
    bit<32> i;
}
```

Source Code 3: metadata definition

In this example, the variable i is stored inside the metadata as shown in Source Code 3. The the variable is annotated with the constant RECIRC_FL (constant with the value 0).

```
apply {
    if(standard_metadata.instance_type != 4)
    {
        meta.index = 0;
    }
     if (!(meta.index < CICLES) && hdr.ipv4
  .isValid()) { //condition check
        //if break condition is met the
  tables for routing need to be applied
          ipv4_lpm.apply();
     }
  }
```

Source Code 4: Ingress control

In the Ingress control, the first thing that is beeing done is to check if the packed arrives here for the first time or if it is a packet that was already recirculated. This is done by checking the field instance_type of the standard_metadata. If the field does not contain the value four [5], it is not a recirculated packet, and the metadata fields need to be initialized. In this example, the variable i gets set to 0 in order to keep track of the number of iterations the packet has already done. If the packet that arrived in the Ingress control is recirculated, the condition will be checked to make sure if the packet needs continue looping or if the loop should be stopped and the packet should be routed.

```
apply {
    if(meta.index < CICLES) { //condition
  check
        //loop-body
        meta.index = meta.index + 1; //
  condition-update
        recirculate_preserving_field_list(
  RECIRC_FL);
    }
  }
```

Source Code 5: Egress control

When entering the egress control, the break condition of the loop will be checked. If the condition holds, the execution of the loop body will start. At the end of the execution, the recirculate_preserving_field_list() method is used. The reason why the recirculate_preserving_field_list() method is used inside the if statement is because otherwise, the routing done by the ipv4_lpm.apply() in Source Code 4 would be overwritten, and the packet would continue looping. Like mentioned before, the recirculate extern takes an integer value in order to save the annotated variables inside the metadata. Here the constant RECIRC_FL is given to the method to save the value of the variable i. After the Egress control, the packet will be reassembled by the deparser and sent back to the parser.
One problem with this approach of building a loop in $P4_{16}$ is that it will create a large overhead for each iteration since it has to pass through the whole package processing every iteration. Also, this solution is unique to the V1 architecture, or other targets that implement a recirculate extern function. [6]

3.3. Custom Loop Header

This solution is similar to the loops introduced in Section 3.2. Instead of relying on an extern method for recirculating, a physical connection between one of the output ports to one of our input ports of the device is beeing established, in order to recirculate the packets. Also a personalized header struct is introduced where all variables used inside the loop will be stored. This means variables needed for the calculations and the variables needed for the break condition check will be saved inside this header. Once a packet arrives, all headers of the packet will be parsed, including the custom loop header. After the parsing is done the program would need to check in the Ingress control if the packet headers include the custom loop header or not. If the packet does not arrive with the custom header that header will be generated and added to the packet headers. If the packet headers already include the custom header it means the packet is looping and the break condition needs to be checked. If the condition is met and the loop needs to be continued the loop body is executed. After the execution of the loop body the variable defining the output port needs to be updated to the port connected to with one of the input ports. If the loop needs to be stopped the custom header needs to be removed again and the packet will be routed. This approach comes with the same problem of the recirculate loops. Since the packet needs to go through the whole packet processing for each iteration, the loop overhead is quite extensive. An advantage of about this approach is that it can be run on every P4 programmable device since the device has no idea it is sending a packet back to itself.

4. Loop Performance Analysis

The test setup that was beeing used for the performance analysis is based on a mininet simulation for $P4_{16}$. The Mininet setup is running on a Virtual machine using Virtual Box. It was set up with the repository [7] and vagrant. In the Virtual Box settings, the VM had four cores and 4096 MB of RAM assigned. The Computer the tests were run on is a laptop with an Intel Core i7-9750H (2.6Ghz) and 8GB of 2667MHz RAM. The mininet topology used has two hosts, h1 and h2, connected with a switch s1 in between. The P4 Program is run on the switch s1. The solution shown in Section 3.3 will not be coverd in the performance analysis. The simulation of the physical connection from one of the outgoing port to one of the ingoing ports was attempted but not succesful. Further studies could try to either verify the solution presented in the section 3.3 outside of a simulation on a real device or find another solution inside the simulation. Because of this, the solution from section 3.3 is only a concept but is not verified if it would acually work.

Figure 2: Mininet topology [8]

To get the measurements shown in Figure 3 below, 50 identical packets were sent from host h1 to host h2 and computed an arithmetic mean of the measured times. The time it took to execute the loop inside switch s1 was calculated by capturing the pcaps of the input and output interface. Pcaps are files that are generated by sniffing on an interface of a network device and they store information like the time a packet left the interface or the IP adress of the packet. Since this Paper focuses on the performance of the different loop types no calculations were done inside the loops to compare just the performance of the loop concepts. As seen in Figure 3, six measurments were beeing done for each approach, starting from 5000 loop iterations to 50000 loop iterations. It was observed that the first packets that were sent had very long times. Because of that the decision was made to send ten packets before measuring the time of the 50 packets used to calculate the average. Another observation that was observed is that the times of most packets were pretty close, but a few of the packets had high variances. This could be a scheduling problem since the p4 program is run inside a mininet simulation inside a VM.

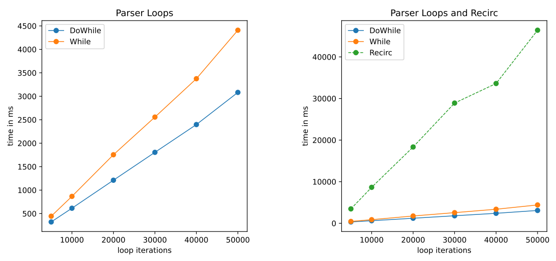


Figure 3: execution time graphs

As expected, the performance of the loops implemented in the parser is better than the one using the recirculate_preserving_field_list() method. This is expected because the recirculate loop has to do the whole package processing for each loop. The performance of the do-while-loop is close to the performance of the while-loop but the do-while-loop is still the faster of the two. This should be because the do-while-loop only uses one state instead of two. The difference between the do-while-loop and while-loop is pretty small and will get even smaller once more states are introduced to the loop for realizing if statements with the select statement. Another thing that can be seen very clearly in the plot is that the time grows linearly the more iterations are added to the loop. This is good to see because otherwise, it would not be suitable for algorithms that need a high iteration count.

5. Conclusion

We saw it is possible to write loops in $P4_{16}$, with some solutions being more costly than others. It can be said that loops used in the parser are a better solution than the recirculate loop. Unfortunately, it was impossible to compare our solutions to the approach mentioned in Section 3.3. One question that remains unanswered is whether it is a good idea to use loops in $P4_{16}$ in general. To answer this question, it would be necessary to do more studies that use the concepts introduced in this paper to implement algorithms for real-world use cases and compare them to solutions from other languages.

References

[1] M. Budiu and C. Dodd, "The p416 programming language," *ACM SIGOPS Operating Systems Review*, vol. 51, no. 1, pp. 5–14, 2017.

[2] D. Scholz, S. Gallenmüller, H. Stubbe, and G. Carle, "Syn flood defense in programmable data planes," in *Proceedings of the 3rd P4 Workshop in Europe*, 2020, pp. 13–20.

[3] B. N. Vladimir Gurevich, "P416 Introduction," https://opennetworking.org/wp-content/uploads/2020/12/p4_d2_2017_p4_16_tutorial.pdf, 2017, [Online; accessed 25-September-2022].

[4] T. P. L. Consortium, "P416 Language Specification," https://p4.org/p4-spec/docs/P4-16-v1.2.3.pdf, 2022, [Online; accessed 25-September-2022].

[5] p4language, "intrinsic," https://github.com/p4lang/p4c/blob/main/testdata/p4_14_samples/switch_20160512/includes/intrinsic.p4#L60-L66, 2019, [Online; accessed 25-September-2022].

[6] A. Fingerhut, "v1model special ops," https://github.com/jafingerhut/p4-guide/tree/master/v1model-special-ops, 2021, [Online; accessed 25-September-2022].

[7] p4lang, "tutorials," https://github.com/p4lang/tutorials/, 2022, [Online; accessed 25-September-2022].

[8] "Working with P4 in Mininet on BMV2," https://opennetworking.org/wp-content/uploads/2020/12/p4_d2_2017_p4_16_tutorial.pdf, [Online; accessed 25-September-2022].

TSN Qbv and Schedule Generation Approaches

Vishavjeet Ghotra, Max Helm*, Benedikt Jaeger*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: vishav.ghotra@tum.de, helm@net.in.tum.de, jaeger@net.in.tum.de

Abstract—**Ethernet is one of the most widely used LAN technologies. This vast application of Ethernet in different fields is due to its features like low cost, flexibility, reliability and easy maintenance. Some application fields like industrial automation require deterministic networks but standard Ethernet is not designed to meet real-time requirements. Time Sensitive Networking (TSN) is a set of standards designed to provide determinism and enable real-time capabilities in Ethernet-based networks. IEEE 802.1Qbv is one of these standards that uses Time-Aware Shaper (TAS) to create Time Division Multiple Access (TDMA) schemes for scheduling and transmitting data in a deterministic manner. This paper provides an overview of IEEE 802.1Qbv and TAS, as well as explains the purpose of schedule generation and compares three different approaches to generate schedules.**

Index Terms—**time-sensitive networking, time-aware shaper**

1. Introduction

Ethernet is a popular network technology known for its low maintenance requirements and high data transfer rates. However, it does not provide the low latency needed for certain fields such as automotive and network automation that require real-time and deterministic communication [1].

To solve this problem, *Time Sensitive Networking* (TSN) standards were introduced in 2012 to provide standardized real-time mechanisms across Ethernet networks. Some of these standards are IEEE 802.1AS (provides clock synchronization), IEEE 802.1Qbu (frame preemption), IEEE 802.1Qbv (enhancements for scheduled traffic), and IEEE 802.1Qca (Path Control and Reservation). In IEEE 802.1Qbv, incoming frames are assigned to different queues on basis of their traffic class. A gate mechanism called *Time Aware Shaper* (TAS) defines a timed gate for each queue that opens or closes in accordance with the schedule implemented in form of *Gate Control List* (GCL). The GCL contains the configuration of all timed gates (open or closed) and determines which gate should be opened for transmission in a given time slot.

Because IEEE 802.1Qbv does not define any algorithm for schedule synthesis, several algorithms are proposed by researchers that focus on different network configurations and traffic classes. This synthesis problem can be reduced to NP-hard problems like bin-packing [2] and thus, itself is also an NP-hard problem.

The goal of this work is to explain the working of IEEE 802.1Qbv (TAS) and explain the working of three approaches for synthesizing schedules. The remainder of this paper is structured as follows. In Section 2, we introduce relevant research on schedule generation. Section 3 provides information about the working of IEEE 802.1Qbv and categorizes different types of methods for creating schedules. Afterward, we describe three schedule approaches in Section 4. Then we evaluate and compare these approaches in Section 5. Lastly, we conclude in Section 6.

2. Related work

As we delve into the workings of 802.1Qbv TAS, it is important to consider the contributions of prior research on schedule synthesis. In the following section, we review relevant literature in this field.

Steiner [3] introduced an Satisfiability Modulo Theory (SMT) based approach to schedule Time-Triggered (TT) traffic in a network. He formulated the scheduling problem as a set of logical constraints that could be solved by SMT solvers. Hellmanns et al. [4] proposed a hierarchical approach that uses a Tabu Search algorithm to schedule large factory networks. Dürr et al. [5] developed a no-wait scheduling algorithm based on Integer Linear Programming (ILP) and Tabu Search. Berisa et al. [6] presented a heuristic method for improving the schedulability of Audio Video Bridging (AVB) streams

In this work, we first survey the SMT based approach proposed by Craciunas et al. [7] to schedule Scheduled Traffic (ST) frames. Next, we explore the method introduced by Houtan et al. [8] for generating schedules that enhances the Quality of Service (QoS) of best-effort traffic in TSN networks. Lastly, we review the window-based heuristic approach proposed by Reusch et al. [9] to schedule large networks.

3. Background

In this Section, we explain the gate mechanism introduced in 802.1Qbv and the concept of the GCL. Afterward, we introduce two types of approaches for generating schedules.

3.1. Time-Aware Shaper

To enable scheduling of Ethernet frames using IEEE 802.1Qbv standards in time-sensitive networks, network components such as switches that are compatible with Qbv are used. Traffic in a network is classified mainly into

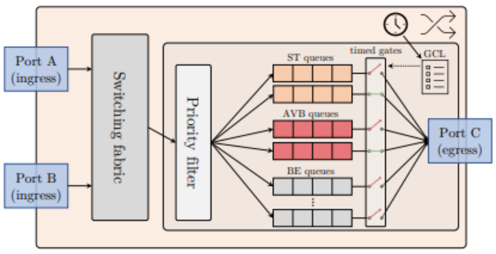


Figure 1: Switch with IEEE 802.1Qbv [6]

three classes: ST, AVB, and Best Effort (BE). ST streams have the highest priority and demand high determinism, whereas BE streams have the lowest priority.

Figure 1 presents a TSN-compatible switch with 2 ingress ports A and B and one egress port C. The function of the switching fabric is to perform mapping of these ingress ports to egress ports for a given stream. There are 8 queues assigned to 3 traffic classes. ST traffic is assigned to higher-priority queues and lower-priority queues are dedicated to BE traffic. The priority filter decides in which queue a frame of the given stream will be enqueued. TAS introduces a timed gate in each queue. Each timed gate has two states: open (1) or closed (0). Frames are dequeued from a queue in a FIFO manner. If gates of multiple queues are opened simultaneously, the frames from the highest-priority queue are transmitted. The state of each timed gate and the time of opening/change in the state of these gates is encoded in a GCL [6]. Each entry in the GCL consists of 2 elements, where the first element is the time relative to the start of the GCL and the second is the state of the gates represented as a bitmask. For example, the entry "T_1: 10000000" means that only the gate of the first queue is opened at the relative time T_1. GCL is a cyclic schedule, i. e., the entries in GCL repeat themselves after a predefined time period T_{cycle}. Furthermore, the IEEE 802.1AS standard is used to synchronize the clocks of all TSN switches in the network [5].

TAS defines the whole gate mechanism that allows or denies the transmission of frames by opening or closing the corresponding queues on the basis of the GCL, but it does not define how the entries in the GCL are created, i. e., it does not define an algorithm that decides the state of the gates at a given time. Therefore, several researchers proposed algorithms that allow optimal communication for different parameters, e. g. some algorithms aim to achieve minimum latency for ST streams without taking the latency of lower-priority streams into consideration [7] while some tend to minimize the maximum end-to-end delay of BE streams [8].

3.2. Schedule Creation

The approaches for creating schedules can be grouped into two categories: exact approaches and heuristic approaches. Exact approaches use methods like SMT, Optimization Modulo Theorem (OMT), and ILP. In such approaches, a constrained satisfaction problem is constructed from the scheduling problem and then solved by methods like ILP or SMT. One of the main advantages of these approaches is that the results generated from these approaches are provably optimal. But the scheduling problems are NP-hard, thus calculating exact solutions for large networks demands high computation time.

Heuristic approaches solve this problem and speed up the calculations by finding sub-optimal solutions. But the results generated by these approaches are not provably optimal. Many of these approaches employ heuristic techniques such as Tabu Search and Simulated Annealing.

4. Approaches for schedule creation

In this Section, we explore three approaches for schedule creation. First, we explore the SMT-based approach to schedule ST traffic by Craciunas et al. [7]. Afterward, we analyze another SMT-based approach to improve the QoS of BE traffic proposed by Houtan et al. [8]. Finally, we inspect the window-based approach devised by Reusch et al. [9] to schedule traffic in large networks.

4.1. SMT Based ST Scheduling

Craciunas et al. [7] formulated the scheduling problem as a set of constraints. These constraints are then passed to the SMT solvers to find the optimal values for variables like frame offset such that all constraints are satisfied.

Many factors like sharing of the same queue by frames of different streams affect the deterministic behavior of a network. To avoid this, the following constraints were defined in [7].

Frame Constraint. This constraint assures that offset of each frame is greater than or equal to 0 and less than or equal to the frame period and thus, guarantees that the frame transmission will be completed before the start of the next period. [10].

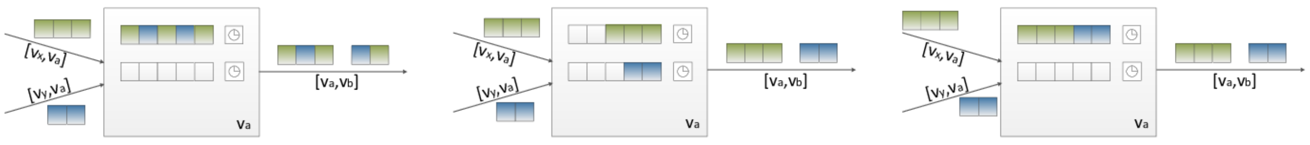
Link Constraint. Only one frame can be transmitted at a time on a given physical link. This constraint enforces that no two frames directed to the same physical link overlap each other [10].

Flow Transmission Constraint. This constraint enforces that frames of a stream follow the routed path of the stream in a specific order. In other words, it assures that a frame can only be sent on the next link in the path after it has been fully received on the previous link [7].

End-to-End Constraint. The time between a stream being transmitted by the sender and received at the destination must be less than or equal to the specified duration to avoid any deadline misses for ST streams [7].

If frames of two different streams arrive at the same time, the order in which the frames are placed in queues is not clear. As shown in Figure 2(a), the frames from both flows may get enqueued in any order and therefore, they may be interleaved in any combination on the egress port. Conditions must be defined to avoid this interleaving by either placing such frames in different queues (Figure 2(b)) or maintaining the intended order and transmission time for all frames if the streams are placed in the same queue (Figure 2(c)) [7].

Stream Isolation Constraint. Two frames `F1` and `F2` of different flows are planned to arrive and be assigned to a queue in a particular order, e. g., `F1` before `F2`. If frame `F1` is lost and is not attached to the queue, frame `F2` will get transmitted in the time slot reserved for `F1`, which



(a) Interleaving streams in egress queue. (b) Isolation of streams in different queues. (c) Isolation of stream arrival.

Figure 2: Flow interleaving and isolation within an egress port [7]

results in non-determinism [7]. To avoid this problem, this constraint enforces that if a frame of a given flow is queued, no frame of other flows is allowed to enter the queue until all frames from the given flow are fully transmitted to the output port [7].

Frame Isolation Constraint. The constraint specified above decreases the search space for the schedules. This new constraint loosens it and enforces that the frames from the second flow are now only required to wait for the frames present in the queue and not for *all* frames of the first flow to be dispatched to the output port [7].

The concept of using Satisfiability Modulo Theories (SMT) solvers to create schedules was first introduced by Steiner [3]. SMT solvers provide a model for the given context when a set of variables and constraints in the context is satisfiable. This model represents one of the potentially many solutions to the given constraints. The goal of this 802.1Qbv scheduling algorithm is to find the optimal values for the frame offsets and queue assignments for each egress port of routed flows in the network such that all constraints explained above are satisfied. This approach involves scheduling each flow one at a time by adding the flow's variables and constraints to the SMT context and attempting to solve the problem [7].

If a solution is found, the variables for the flow are replaced by the constant value provided by the SMT model. This repeats until all flows are scheduled (a solution found) or the solver determines that it is not possible to satisfy newly added constraints in the context. If an unfeasible step is encountered, the SMT context is backtracked, and the last added flow is removed. The next optimal values of variables for the removed flow are then determined before returning to the unfeasible step. This backtracking algorithm repeats until a complete solution for all flows is found or it is found that solution does not exist after checking every possible combinations of values using backtracking. In the latter case, a suboptimal solution is returned that is able to schedule the maximum number of flows [7].

4.2. QoS Improvement

The approach explained above addresses the scheduling of ST streams but does not focus on BE traffic class. The approach in [8] is an enhancement that uses all constraints defined in [7] and proposes a new set of constraints and objective functions to create schedules for ST traffic while improving QoS for BE traffic.

The concept of slack is introduced to accommodate BE frames between consecutive ST frames. Slack is a period of time after the transmission of an ST frame during which no other ST frames can use the bandwidth of the link [8].

Porous Link Constraint. This is a modified form of the link constraint explained above. The link constraint avoids the timely overlap of frames on the same link. This constraint has been modified to take slack into account and avoids overlapping of frames along with their slacks [8].

Slack Size Constraint. This constraint verifies the size of slack used for each frame scheduled on the link. The used slack size must be greater than or equal to zero, but it must also be less than or equal to the difference between the frame period and its transmission time [8].

Hop Slacks Constraint. This constraint bounds the total amount of slacks allowed on the link [8].

Equal Link Constraint. This is an optional constraint that enforces equal slack sizes for all frames on a link [8].

Three objective functions were introduced in addition to these constraints.

Maximization. This optimization objective function attempts to schedule the transmission of ST frames close to their deadlines, while still packing them together. This maximizes the allocation of available bandwidth for the transmission of BE frames at the start of the schedule and avoids deadline misses for these frames [8].

Sparse Schedule. This function seeks to maximize the overall slack between subsequent ST frames that are scheduled to be transmitted on the same link. In other words, it helps to create gaps between subsequent ST frames to fit BE frames to minimize the delay [8].

Evenly Sparse Schedule. This function modifies the sparse schedule objective function by asserting the equal porous constraint specified above to create equally-sized slacks on the links [8].

An SMT solver checks the satisfiability of these constraints and returns one solution if they are satisfiable.

4.3. Window-Based Heuristic Approach

Real-time communication is critical in large industrial networks. SMT-based approaches are not ideal for scheduling these large networks, because the large constrained problems formulated have exponential runtime w. r. t. the number of flows. Thus, heuristic approaches are widely used for scheduling large networks, as they provide solutions in a shorter period of time that may or may not be optimal.

Reusch et al. proposed a heuristic approach in [9] to create schedules. The main goal of this work is to find the optimal window length and period so that no deadline for an ST flow is missed and the total available bandwidth of all ports in the network is maximized. The purpose of maximizing available bandwidth is to allow room for lower-priority traffic.

A window of a queue refers to the time interval for which the timed gate of the queue is opened and allowed to transmit frames. A cost function is defined for each port that determines the number of windows active in a given period. These cost functions are added together to get the overall cost function. A heuristic algorithm is developed to find an initial solution that minimizes this overall cost function. A solution is considered *valid* if all flows have a finite worst-case delay. If the initial solution provided by the heuristic method is *valid*, it might be cost-optimal but it does not imply that all deadlines are met. Thus, a window optimization algorithm checks if the initial solution satisfies the deadlines of all flows. If these deadlines are met, then the initial solution is returned as the final solution.

Otherwise, the infeasible flows are sorted in descending order based on the percentage by which they exceed their deadlines. The flow with the largest percentage is optimized first. Optimizing a flow involves adjusting the period of all ports on its route through an iterative process. This is done by increasing or decreasing the period of every window in each port by the same amount in each iteration. If the flow is not feasible after adjusting the periods, they are decreased for all ports. If the flow is feasible, the periods are increased until the solution becomes infeasible or the periods no longer change. The motive here is to find the maximum period for each port that allows all flows to meet their deadline [9].

Ultimately, we either get a feasible solution or there could be a solution that exists but cannot be discovered through this method.

5. Comparison

In this section, we discuss the results achieved by the three approaches. We present and compare the results obtained using these approaches.

5.1. ST Scheduling

Z3 SMT/OMT solver and Yices v2.4.2 (64bit) were used for evaluation. The experiments were conducted on a 64bit 4-core 3.40GHz Intel Core-i7 PC with 4GB memory and a 5-hour time-out value. Synthetic configurations were used to analyze the scalability and schedulability of the networks, based on three predetermined network topologies. These topologies ranged from 3 end-systems connected to one switch to 7 end-systems connected through 5 switches. To achieve higher utilization on the links, the size of the topologies was kept relatively small compared to the number of flows. The results indicated that the runtime increased exponentially with an increase in the number of flows and frames scheduled, while the period set has a significant impact on scalability. The flow isolation approach consistently outperformed the frame isolation approach, with an average 13% faster runtime at the expense of schedulability. Further experiments confirmed this trend, with the flow isolation approach being faster on average than the frame isolation approach [7].

5.2. QoS Improvement

For the purpose of evaluation, a multi-hop network with six end stations was used. Different scenarios were selected that featured different ratios of ST and BE streams. In addition to the three objective functions explained in the Section 4.2, the minimization objective function is used to schedule ST streams. The minimization objective function minimizes the total offset of ST streams. The end-to-end delay of BE streams, deadline misses, and runtime were measured for each objective function. Moreover, Z3 SMT/OMT solver was used to solve the constrained problem and find a feasible solution.

The minimization objective function resulted in the longest end-to-end delay for BE streams in all scenarios, while the maximization objective function gave the best results. The reason is that the maximization function schedules ST frames as close to their deadlines as possible, which creates room for BE frames to be scheduled. The sparse and evenly sparse objective functions also produced lower end-to-end delays due to the slack, that is built in to accommodate BE frames.

The minimization and maximization functions had a large number of missed deadlines, while the sparse schedule function and evenly sparse schedule functions had no missed deadlines in any of the scenarios. Both the sparse schedule function and evenly sparse schedule function also had the best runtime. In scenarios with fewer ST streams, the maximization function was faster than the minimization function. However, as the number of ST streams increased, the maximization function became slower and was eventually outperformed by the minimization function [8].

5.3. Window-Based Heuristic Approach

Seven test cases were designed based on industrial application requirements to evaluate the effectiveness of the approach. Moreover, the greedy randomized adaptive search procedure (GRASP) metaheuristic from [11] and Strict Priority (SP) policy were used for comparison. Four aspects were considered in the comparison: worst-case delay, mean cost, calculation time, and the number of infeasible flows.

The results showed that although the proposed algorithm had a better execution time than GRASP, it was outperformed by GRASP in the other three aspects. The SP algorithm had the best runtime, but it caused a significant delay for other traffic classes. GRASP was able to schedule all flows in all test cases, but required exponential time to get solutions in some cases. Moreover, GRASP was less robust and more difficult to adapt to changing conditions in real-time applications, compared to the proposed algorithm [9].

5.4. Comparison

As shown in the Table 1, ST Scheduling [7] solely focuses on scheduling ST frames, while the other two approaches take both ST and BE frames into consideration. Additionally, ST Scheduling [7] has the highest runtime among the three approaches. For small networks, QoS Improvement [12] exhibits the most efficient runtime, whereas Window-based Heuristic [9] has the best runtime for larger networks but as a tradeoff, it returns suboptimal solution, unlike the other two approaches that always return optimal solution.

Therefore, we can conclude that out of the three approaches, the third approach is optimal for large networks,

Name	Solution Approach	ST	BE	Nodes	Streams	Runtime (s)	Solution Optimal?
ST Scheduling [7]	SMT	✓	×	12	1000	<18000	✓
QoS Improvement [12]	SMT	✓	✓	8	10	0.37-1153.52	✓
Window-based Heuristic [9]	Heuristic	✓	✓	402	316	∼10.52	×

TABLE 1: Overview of the approaches discussed in this paper [13]

whereas for small networks, the second approach is the most suitable approach.

6. Conclusion and Future Work

In this paper, we described the working of the gate mechanism in IEEE 802.1Qbv and explored and analyzed three different approaches for creating schedules. The first approach scheduled ST frames without taking other traffic classes in consideration. The second approach aimed to optimize the scheduling of BE frames while meeting the deadlines of ST frames for small networks using SMT solvers. The last approach used a heuristic algorithm to schedule ST frames and maximize the available bandwidth for BE frames in large networks. All approaches succeeded in their objectives. We also compared these approaches and concluded that the third approach had the best runtime at the expense of the optimality of the solution returned, whereas the second approach outperformed the first approach for small networks.

Future work on the first and second approaches could focus on reducing the complexity of the scheduling problem by trying to remove or loosen some constraints while still meeting the deadlines of all flows. One possible extension to the window-based approach could be further optimization using new heuristic techniques to minimize the overall delay.

References

[1] “Is Ethernet Deterministic?” https://polytron.com/blog/is-ethernet-deterministic-does-it-matter-part-1/, [Online; accessed 19-December-2022].

[2] A. A. Syed, S. Ayaz, T. Leinmüller, and M. Chandra, “Mip-Based Joint Scheduling and Routing With Load Balancing For tsn Based In-Vehicle Networks,” in *2020 IEEE Vehicular Networking Conference (VNC)*. IEEE, 2020, pp. 1–7.

[3] W. Steiner, “An Evaluation of SMT-based Schedule Synthesis for Time-Triggered Multi-Hop Networks,” in *2010 31st IEEE Real-Time Systems Symposium*. IEEE, 2010, pp. 375–384.

[4] D. Hellmanns, A. Glavackij, J. Falk, R. Hummen, S. Kehrer, and F. Dürr, “Scaling TSN Scheduling for Factory Automation Networks,” in *2020 16th IEEE International Conference on Factory Communication Systems (WFCS)*. IEEE, 2020, pp. 1–8.

[5] F. Dürr and N. G. Nayak, “No-Wait Packet Scheduling for IEEE Time-Sensitive Networks (TSN),” in *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, 2016, pp. 203–212.

[6] A. Berisa, L. Zhao, S. S. Craciunas, M. Ashjaei, S. Mubeen, M. Daneshtalab, and M. Sjödin, “AVB-aware Routing and Scheduling for Critical Traffic in Time-sensitive Networks with Preemption,” in *Proceedings of the 30th International Conference on Real-Time Networks and Systems*, 2022, pp. 207–218.

[7] S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner, “Scheduling Real-Time Communication in IEEE 802.1 Qbv Time Sensitive Networks,” in *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, 2016, pp. 183–192.

[8] B. Houtan, M. Ashjaei, M. Daneshtalab, M. Sjödin, and S. Mubeen, “Synthesising Schedules to Improve QoS of Best-Effort Traffic in TSN Networks,” in *29th International Conference on Real-Time Networks and Systems*, 2021, pp. 68–77.

[9] Reusch, Niklas and Zhao, Luxi and Craciunas, Silviu S. and Pop, Paul, “Window-based schedule synthesis for industrial ieee 802.1qbv tsn networks,” in *2020 16th IEEE International Conference on Factory Communication Systems (WFCS)*, 2020, pp. 1–4.

[10] S. S. Craciunas and R. S. Oliver, “Combined Task-and Network-Level Scheduling for Distributed Time-Triggered Systems,” *Real-Time Systems*, vol. 52, no. 2, pp. 161–200, 2016.

[11] M. L. Raagaard and P. Pop, “Optimization Algorithms for the Scheduling of IEEE 802.1 Time-Sensitive Networking (TSN). Tech. Univ. Denmark, Lyngby,” Denmark, Tech. Rep, Tech. Rep., 2017.

[12] S. Martello and P. Toth, “Bin-packing Problem,” *Knapsack problems: Algorithms and computer implementations*, pp. 221–245, 1990.

[13] T. Stüber, L. Osswald, S. Lindner, and M. Menth, “A Survey Of Scheduling In Time-Sensitive Networking (TSN),” *arXiv preprint arXiv:2211.10954*, 2022.

Reproducible Network Experiments using NixOS

Michael Hackl, Kilian Holzinger*, Henning Stubbe*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: michael.hackl@tum.de, holzinger@net.in.tum.de, stubbe@net.in.tum.de

***Abstract*—This is a technical report about making NixOS available on the Chair of Network Architectures and Services' testbeds. Our goal is to make it easy for conductors of network experiments to make their experiments reproducible. NixOS was chosen for its reproducible and declarative system and package management. In the end, we will have integrated NixOS into the chair's testbed infrastructure and conductors can choose to use a ready-made image of NixOS for their experiments.**

***Index Terms*—reproducibility, experiments, testbed, pos, NixOS**

1. Introduction

An important part of science is the verifiability of results. To promote verifiability, we want to help conductors make their experiments on the chair's testbeds more reproducible, i. e., the experiments can be replicated and will still have the same outcome.

While the reproducibility of an experiment has many facets, in this paper, we focus on the operating system: different researchers should be able to set up their machines to the same operating system state that the conductor had when he or she performed the experiment. There are two parts to this operating system state: the installed programs and the configuration. Both parts are dealt with in this paper.

For this reason, we will continue Zhou Lu's previous Bachelor's thesis "Reproducible Research Infrastructure with NixOS" [1], [2] by

- first giving an overview of the environment where the experiments take place, i. e., the testbeds of chair I8, and outlining NixOS from a reproducibility perspective,
- then showing how to make NixOS available on testbeds using pos,
- and finally thoroughly describing the implementation details of this process.

2. Background Information

All of the following information about the Chair of Network Architectures and Services' (I8) testbeds is from its wiki page [3].

2.1. Testbed Machines

Each testbed of the I8 testbeds consists of a management node and test nodes. Every authorized user can connect to the management node via SSH and, as the name suggests, manage the test nodes from there. The test nodes are bare-metal servers on which the experiments can be executed. Some are connected among each other with specified network links, over which experiments can be run. They do not have an operating system installed on them.

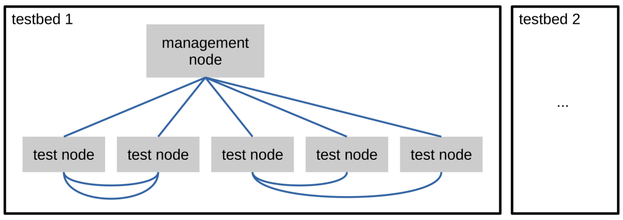


Figure 1: A simplified representation of the structure of the testbeds.

2.2. pos

The management and test nodes work together with the help of pos [4]. pos, which stands for "Plain Orchestrating Service", is the main tool for conducting experiments on the testbeds. It is used for

- managing the access to the test nodes between the users through allocations and reservations with a calendar,
- managing test nodes in terms of powering the test nodes on or off and providing them an operating system to boot,
- configuring test nodes with the parameters of an experiment,
- conducting an experiment on test nodes by executing a script or individual commands on them,
- and gathering the results and artifacts of an experiment.

pos consists of three main components:

- The *pos daemon (posd)* is running on the management node and manages the experiments. It provides a REST (representational state transfer) API for the other components.
- posd can be controlled with *poslib and pos-cli* from the management node. poslib is a python library and pos-cli is a command line interface. pos-cli uses poslib internally.
- *postools* is available on the test nodes and does the communication with posd, e. g. synchronizing

the test nodes with checkpoints and taking care of other common useful tasks for experiments. postools can be used as a python library or as a command line interface.

2.3. Operating System Images

The management node can control the test nodes per IPMI (Intelligent Platform Management Interface). This interface allows posd to do low-level (independent of an operating system) tasks such as powering the machines on. The test nodes then boot by means of network booting (Preboot eXecution Environment, PXE), i. e., they use files from the network instead of from a built-in storage. These files are provided by pos. pos allows us to choose the operating system which it supplies the test nodes with, e. g. Debian Bullseye. The operating systems, however, are so-called live boot operating systems, i. e., they do not need to be installed before they can be used.

New operating systems, such as NixOS in our case, can be added to pos in the form of images. An image is a directory that contains the following files:

- `vmlinuz`: the Linux kernel,
- `initrd.img`: the initial root file system image,
- and (optionally) `bootparameters.yml`: the kernel boot parameters that pos should add[1].

Such an image needs to be stored in `/srv/testbed/images/staging` on the management node, which is the place where every user is allowed to add new images.

The standard way of creating images for the I8 testbeds is using mandelstamm [6], a collection of scripts, on the so-called builder image. It supports different operating systems via a module system. The main script `build.sh` executes the desired module (which is just another script) and then packs the result into an image.

2.4. Bootstrapping

After a test node boots, it needs to be set up. pos does this by executing the python program `host.py` [7] on the test node through SSH. It sets the configured environment variables, installs postools, and updates the SSH keys.

This program is not compatible with NixOS as is and needs to be adapted to work correctly.

2.5. Motivation for the Choice of NixOS

There are already several versions of Debian and Ubuntu available for use as an operating system on the test nodes. Also, by using network booting, the test nodes do not store state between the experiments and therefore already incentivize making experiments reproducible. In this section we explore why it is worth it nonetheless to make NixOS available on the testbeds and what it can improve on the current situation.

NixOS [8] is a Linux distribution based on the Nix package manager. Both the system configuration (e. g. the `/etc` directory) and the package management, are handled in a purely functional language called Nix expressions. These Nix expressions describe derivations, which are tasks that define everything that is needed in order to build a package. The Nix Packages collection (nixpkgs) [9] contains Nix expressions for many commonly used packages. By specifying the git revision of the Nix Packages collection repository, the versions of all packages are clearly defined.

To install a package, the corresponding derivation has to be realized. The output is then stored in a central place: the Nix store. When some packages are not needed anymore, e. g. old versions after an update, the Nix store can be cleaned of old and unused packages by calling the garbage collection.

NixOS uses source-based package management with a binary cache. That means that in contrast to, for example Debian, whose packages are distributed as binaries, its packages are distributed in source form. However, to save build time, NixOS can download pre-built binaries from a binary cache when the inputs of the derivation match the inputs of the cached version. This model is great for reproducibility because in case a package is not available anymore for download in the future, Nix can automatically build it again as long as its source code can be found. Since the Nix expressions are (supposed to be) deterministic, the resulting binary is identical.

The system configuration of NixOS is declaratively defined in the file `/etc/nixos/configuration.nix`. A changed configuration can be applied with the program `nix-rebuild`. This means that one has to share just this one file (and the files that are referenced from it) and others can reproduce the whole system configuration (kernel, system services, applications, configuration files, etc.) except for mutable state (e. g. the `/var` directory). This is more efficient and less error-prone than using shell scripts and manual commands to configure the system.

3. Using NixOS in pos Testbeds

The general command to specify which operating system image to use is

```
pos nodes image <node> debian-bullseye
```

when using one of the provided images, e. g. Debian Bullseye. To apply this choice, we subsequently need to restart the test node:

```
pos nodes reset <node>
```

If we want to use a self-made image, we need to add the staging argument to our above command:

```
pos nodes image --staging <node> <image>
```

But we may not need to build the NixOS images ourselves: mandelstamm-ci [10] is a program that builds images for the testbeds using mandelstamm at predefined intervals. In case we want to build the image ourselves anyway, e. g. because we want to change the configuration for the image beforehand or mandelstamm-ci does not build the images for our testbed, we can do it as follows in the “builder-bullseye” image:

```
MANDELSTAMM_TARGET=copy mandelstamm/build.sh
↪  mandelstamm/modules/nixos-22.11.sh <image
↪  name>
```

The NixOS image is already preconfigured like the other images on the testbeds, e. g. useful programs are installed, and the timezone is set correctly.

1. This is currently not disclosed in the wiki [3], but can be read about in an issue of the pos daemon project [5].

4. Implementation Details and Contributions

Our goal is to make NixOS available on the test nodes for experiments.

We base our work on Lu's Bachelor's thesis [1], which explains how to make a NixOS image and provide it to pos and describes some changes to pos and the NixOS image in order to make pos and NixOS compatible with each other so that pos can carry out its tasks, which we cover in Section 4.1 and Section 4.2 respectively.

The following are our contributions: We extend the changes to pos and the NixOS image in Section 4.2. Then we automate the build process of the NixOS image with mandelstamm in Section 4.3 and add it to mandelstamm-ci in Section 4.4. Eventually, we add some guidance on the usage of NixOS in the testbeds' wiki in Section 4.5.

4.1. Building a Basic NixOS Image

Nixpkgs already provides a way to build PXE images of NixOS [8], [9]. In the name of easier reproducibility, we use this official way of creating our NixOS image instead of building it with mandelstamm (though later we will add a feature to mandelstamm that allows external programs—in this case `nix-build`—to build images). This means that our first step is to install the Nix package manager:

```
apt install nix-setup-systemd
```

We use the existing package manager to install Nix instead of piping the contents of the official installer URL to bash in order to save the manual creation of a non-root user account, have better compatibility with Debian, etc. After that, we clone the nixpkgs repository [9], which contains the build instructions:

```
git clone --depth 1
↪  https://github.com/NixOS/nixpkgs.git
↪  --branch nixos-22.11
```

To finally build a NixOS image, we then run:

```
nix-build -A netboot.x86_64-linux
↪  nixpkgs/nixos/release.nix
```

This yields us three files, two of which—`bzImage` and `initrd`—we just need to rename to `vmlinuz` and `initrd.img` respectively to fit as a pos image. The third file—`netboot.ipxe`—needs to be adapted to work with pos. It contains the kernel boot parameters (particularly "init") that are required for booting NixOS. To extract them from this file and write them to the `bootparameters.yml` file, we use the command from Figure A.4 in Lu's thesis [1]:

```
grep --regexp='.*init=.* initrd=initrd.*'
↪  <'netboot.ipxe' | sed 's/.*init=\(.*\)
↪  initrd=initrd.*/init: \1/'
↪  >'bootparameters.yml'
```

4.2. Changes to pos and the NixOS Image

Now the image in itself is done. But pos still needs to be expanded to be able to deal with NixOS and NixOS configured to work together with pos. Some of these changes have already been made (see Figures A.2 and A.3 in thesis [1] and merge request [11]), others we make ourselves (see merge requests [12], [13]).

4.2.1. NixOS Configuration. The NixOS image is assimilated to the existing OS images by including the same configuration that the module `common.sh` applies to all other mandelstamm images in the NixOS configuration. This involves installing likely useful programs, setting the hostname to be received via DHCP (Dynamic Host Configuration Protocol), setting the timezone to "Europe/Berlin", and adding SSH keys.

Furthermore, symbolic links are created in the image to the programs that posd and `host.py` (introduced in Sections 2.2 and 2.4 respectively) expected at certain locations. Python, for example, is linked to `/usr/bin/python`.

We modify the pos daemon to not expect python at a fixed location with `/usr/bin/python` but instead use the environment with `/usr/bin/env python`. The equivalent has already been done for postools [14]. This means that we can remove the creation of the symbolic links from the NixOS configuration.

4.2.2. NixOS Configuration File Location. The corresponding options for the aforementioned configuration are directly included in the base file for the image `netboot-minimal.nix` in the local nixpkgs repository.

This method of changing the configuration of the image has a problem: While the changes in the local repository do carry over to the image (which means that a rebuild retains this configuration), an update (`nix-channel --update`) overwrites them. This means that a subsequent `nixos-rebuild` resets everything.

To solve this, we move the whole configuration to a separate file called `testbed.nix`. We import this file for the build of the image and set that the file is included in the image and referenced from the NixOS configuration file `/etc/nixos/configuration.nix` in the image using the "configuration" argument of `release.nix`:

```
nix-build nixpkgs/nixos/release.nix -A
↪  netboot.x86_64-linux --arg configuration
↪  '{ pkgs, ... }: { imports = [
↪  ../files/testbed.nix ];
↪  installer.cloneConfigIncludes = [
↪  (pkgs.writeText "testbed.nix"
↪  (builtins.readFile ../files/testbed.nix))
↪  ]; }'
```

4.2.3. postools Installation. To accommodate the special way to install software of NixOS, the file `default.nix` [15] containing a Nix expression for building a Nix package of postools is added at `/srv/testbed/files/luz/default.nix` on the management nodes of two testbeds[2]. When bootstrapping a test node, the file is downloaded to the test node and used by `host.py` to install postools to the default profile (available in all user environments).

4.2.4. Hostname Correction. `host.py` needs the short hostname for the communication with pos, but in NixOS the hostname is set to the fully qualified domain name obtained over DHCP, e. g. "klaipeda.baltikum.net.in.tum.de". Therefore `host.py` only uses the part before the first dot of the hostname when bootstrapping NixOS.

2. This file was added to the management nodes "coinbase" and "kaunas" [11]. In the final section of this paper we suggest making this file no longer necessary as future work.

4.2.5. NixOS Identification.
The identification of NixOS in host.py for the decision on how to deal with the hostname and how to install postools fails, i. e., it does not execute the NixOS-specific statements, which causes the bootstrapping process to abort. We fix this by making the comparison that tests for NixOS case insensitive.

4.2.6. Direct Build of bootparameters.yml.
We can produce the bootparameters.yml file directly with Nix instead of adapting netboot.ipxe with sed. To do so, we overwrite the Nix derivation system.build.netbootIpxeScript (in the file netboot.nix [9]) using "mkForce" (explained in [8, Section 67.3.2. Setting Priorities]) to generate our bootparameters.yml file instead of the standard netboot.ipxe file:

```
system.build.netbootIpxeScript = lib.mkForce
↪  (pkgs.writeTextDir "bootparameters.yml" ''
  init: "${config.system.build.toplevel}/init"
'');
```

4.3. Automated Building With mandelstamm

While mandelstamm is not compatible with the way we want to build a NixOS image (with nix-build, see Section 4.1) as is, we will extend it accordingly.

mandelstamm has a feature that allows us to specify with the environment variable MANDELSTAMM_TARGET the format in which it should pack the image. The two options are "traditional" and "squashfs". They were originally introduced to compress images that are too large. We add a third packing format "copy" to these options that just copies the files that a module created to the output directory, i. e., it does not pack them first as the other two options do. This allows for building an (already packed) NixOS image in a mandelstamm module.

As the building of images happens on the builder image, we add "nix-setup-systemd" to the packages to be installed there.

Next, we create the file testbed.nix (with the previously mentioned content, see Section 4.2.1) and a new module for NixOS for versions 22.05 and 22.11 respectively in the mandelstamm repository. These new modules contain the commands for building NixOS that we already discussed. The noteworthy points here are that we do not add the result of the nix-build command as a root of the garbage collector with the option --no-out-link and that we call nix-store --gc at the end to collect the garbage. We do this because, unlike when building the other images, where everything happens in a tmpfs (Temporary File System), there are files left over after building NixOS, namely in the Nix store.

Furthermore, we automatically add the SSH key for pos, which is found in a git submodule of the mandelstamm repository, to the NixOS configuration so that pos is allowed to log in.

Eventually, we bundle all this in merge request [12]. Now NixOS images can be built using mandelstamm in the "builder-bullseye" image using the command shown in Section 3.

4.4. Adding NixOS to mandelstamm-ci

Because we implemented the creation of a NixOS image in mandelstamm and mandelstamm-ci works together well with mandelstamm, we can easily automate the building process with mandelstamm-ci: With merge request [16] we add an entry for our build module to the configuration file of mandelstamm-ci mandelbauer-config.yaml to make it build a NixOS image regularly and we specify the packing format for this build to be "copy".

4.5. Adding Instructions to the Testbeds' Wiki

At the very end, we add instructions for using NixOS in the testbeds to the testbeds' wiki [17].

5. Conclusion

NixOS is now available on the chair's testbeds as an automatically built image and can also be built on other testbeds using pos by following the same procedure. That means that conductors can now easily choose NixOS for their experiments and make their experiments more reproducible this way.

To allow others to reproduce the NixOS environment for their experiments, conductors only have to provide the nixpkgs git revision their system was built with and their configuration.

Last, we suggest some judicious changes that we do not implement but leave for future work:

- Provide a universal way of installing postools so that bootstrapping is possible on all testbeds without needing to manually deploy a file once on each testbed beforehand. This could be done by including the Nix expression for postools directly in the NixOS configuration at the image generation instead of installing postools during the bootstrap process.
- The command nixos-rebuild test applies changes to the system even when using the unchanged configuration file from the image. Find out why this happens and address it.
- The command nixos-version, which shows among other things the git revision the system was built from, reports the dummy values from the release.nix file in [9]. Only after running nixos-rebuild test --upgrade it shows the correct version. The correct values should be provided from the beginning by giving them as arguments to release.nix in the build command.

References

[1] Z. Lu, "Reproducible Research Infrastructure with NixOS," Bachelor's thesis, Technical University of Munich, 2021.

[2] ——, "Accompanying GitLab Project to 'Reproducible Research Infrastructure with NixOS': NixOS for pos," https://gitlab.lrz.de/netintum/teaching/tumi8-theses/ba-lu/nixpkgs, [Online; accessed 4-March-2023].

[3] "Orchestration of Testbeds at I8," https://gitlab.lrz.de/I8-testbeds/wiki/-/wikis/home, [Online; accessed 4-March-2023].

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

[5] “Issue in the pos Daemon Project: Per image (additional) default boot parameter,” https://gitlab.lrz.de/I8-testbeds/pos/daemon/-/issues/209, [Online; accessed 4-March-2023].

[6] “mandelstamm Project,” https://gitlab.lrz.de/I8-testbeds/mandelstamm, [Online; accessed 4-March-2023].

[7] “host.py in the pos Daemon Repository,” https://gitlab.lrz.de/I8-testbeds/pos/daemon/-/blob/master/posd/nodes/boot/host.py, [Online; accessed 4-March-2023].

[8] “NixOS Manual,” https://nixos.org/manual/nixos/stable/index.html, [Online; accessed 4-March-2023].

[9] “Nixpkgs Repository on GitHub,” https://github.com/NixOS/nixpkgs, [Online; accessed 4-March-2023].

[10] “mandelstamm-ci Project,” https://gitlab.lrz.de/I8-testbeds/mandelstamm-ci, [Online; accessed 4-March-2023].

[11] Z. Lu, “Merge Request to the pos Daemon Repository: Nixos support,” https://gitlab.lrz.de/I8-testbeds/pos/daemon/-/merge_requests/323, [Online; accessed 4-March-2023].

[12] “Merge Request to the mandelstamm Repository: Add NixOS,” https://gitlab.lrz.de/I8-testbeds/mandelstamm/-/merge_requests/37, [Online; accessed 4-March-2023].

[13] “Merge Request to the pos Daemon Repository: Increase NixOS compatibility,” https://gitlab.lrz.de/I8-testbeds/pos/daemon/-/merge_requests/399, [Online; accessed 4-March-2023].

[14] “Issue in the postools Project: please use /usr/bin/env in shebangs,” https://gitlab.lrz.de/I8-testbeds/pos/tools/-/issues/24, [Online; accessed 4-March-2023].

[15] Z. Lu, “default.nix in the Accompanying GitLab Project to ‘Reproducible Research Infrastructure with NixOS’: NixOS for pos,” https://gitlab.lrz.de/netintum/teaching/tumi8-theses/ba-lu/nixpkgs/-/blob/master/pos_python_patch/default.nix, [Online; accessed 4-March-2023].

[16] “Merge Request to the mandelstamm-ci Repository: Add NixOS 22.11 to the build schedule,” https://gitlab.lrz.de/I8-testbeds/mandelstamm-ci/-/merge_requests/8, [Online; accessed 4-March-2023].

[17] M. Hackl, “NixOS,” https://gitlab.lrz.de/I8-testbeds/wiki/-/wikis/for-users/testbed-images/NixOS, [Online; accessed 5-March-2023].

Increase the Latency: Emulation Approaches for Real Network Conditions

Lukas Haneberg, Eric Hauser*, Sebastian Gallenmüller*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: ga92nen@mytum.de, hauser@net.in.tum.de, gallenmu@net.in.tum.de

Abstract—**The increasing complexity of network setups in Information Technology fields and companies directly increases the need for constant testing and validation of the network. It is often important for the main network, or the production network, to be running constantly without interruptions, and to be running at full capacity without the additional overhead of testing software. In order to create a testbed that closely resembles the complexity of a production network, network emulation must be utilized. Another way of bridging the problem of missing network accuracy is network simulation, but the findings in this paper suggest that by using a emulation network setup, including actual traffic as opposed to simulated traffic, the production network can be emulated more accurately. In order for a testbed to accurately emulate the real latency behavior of network flow through a real Wide Area Network (WAN), multiple technological approaches can be put to use. This paper compares each of these different approaches and evaluates them based on emulation ability, costs and configuration efforts.**

Index Terms—**network emulation, software link emulators, hardware wan emulators, fiber optic delay lines**

1. Introduction

Testing, validating, and implementing new technologies in computer networks is a core aspect of achieving a solid network infrastructure. The optimal way of doing so is to carry out the testing in the actual production environment, since this is the actual network setup through which all of the live traffic flows, and thus is already setup in the optimal way. But since testing in the production environment can also result in major downsides, such as unwanted crashes and down-times of nodes in the network, this does not present the ideal way of testing or expanding a network [1].

The solution to this problem is to set up a testbed, which is an environment that mirrors or imitates the production network, so that new technologies, expansions in soft- and hardware and general testing of the network devices can be tested. Since a testbed network setup is usually hosted in a single location, this provides the setup with overly ideal conditions. The cable lengths are short, usually staying below 100 m, and there are no additional network nodes, which would be introduced if the network was geographically separated and had to utilize WAN connections. Connections over a WAN can impose delays, latency and other types of interferences simply due to additional network nodes in the network, or propagation delay induced by longer cable routes.

This poses the question of how a testbed can reproduce the characteristics of a live environment, especially over longer distances, in order to make the testing even more accurate. A common answer to this question is network simulation. Network devices are modeled in a virtual environment, which allows for simulation based testing [2]. This is a valid first step in order to test the functionalities and study the behaviors of the network. However, network simulation quickly reaches its limits, since it fails to reproduce the real conditions imposed on the production network.

We propose to use network emulation, which introduces additional factors in terms of actual network devices, while still keeping the valuable aspects of a simulation environment [2]. In order to test the hardware and software, and various types of behaviors within the network over a longer distance, and additional number of devices in the network while still remaining in the local testbed, emulation methods have to be introduced. Traffic over a WAN, for example, introduces latency, packet loss, delays etc. This can be achieved by a variety of methods and techniques, for example a simple Linux device running NetEm, an extension of the already available network manipulation functions of Linux, which can be used to add fixed amounts of delays with additional random latency variation to outgoing packets [3].

This paper focuses on emulation techniques in order to accurately simulate a production environment and analyzes their benefits and limitations in terms of introducing actual network behaviors such as the latency introduced by the cable lengths in WANs. The structure of the paper is as follows: Section 2 lists and compares free software link emulators. Section 3 lists and compares all-in-one hardware WAN emulators. In the following Section 4, a comparison of the functionalities of free software emulators and hardware emulators is made. Section 5 discusses the use of fiber optic delay lines in fiber networks. Finally, Section 6 summarizes and contrasts all of the previously mentioned approaches.

2. Free Software Link Emulators

Software link emulators are software based tools, which require underlying hardware to run on. A PC is enough to install and run software emulators, most of them being integrated into the operating system environment. They are the most commonly used tools for network emulation, as they are usually free and open source [2]. Software emulators can be versatile tools and are useful for network emulation based testing, however their functionalities reach their limits quickly, as high line rates are not realistic and the underlying hardware is not dedicated to the emulation software. In the following sections, the most popular and promising software link emulators are discussed and compared.

2.1. NetEm

NetEm is a free extension to the already existing Linux Traffic Control package. It is used for its emulation functionalities for simulating the characteristics of a WAN, making it an important tool for testing. Command line parameters allow introducing latency to outgoing packets, packet loss, corruption, re-ordering and control bandwidth through rate control. This paper investigates the latency functions of Net Em. Installation is kept simple, when using a Linux kernel 2.6 or higher, it is already enabled. In older kernel versions the implementation is also simple and can be enabled in the Networking Options [3].

Listing 1: NetEm Delay with non purely random variation

```
# tc qdisc change dev eth0 root NetEm
delay 100ms 10ms 25%
```

The command line snippet Listing 1 is taken out of the official NetEm for Linux documentation [3]. The 100 ms parameter adds a fixed delay of 100 ms. The 10 ms adds or subtracts additional 10 ms in a purely random fashion, meaning the outcome of the delay value is 90 ms or 110 ms. Because WAN connections do not usually behave in a purely random fashion, the 25 % percent parameter is necessary to approximate real variation, meaning that the next random element of the delay is dependent on the previous outcome by 25 % [3].

2.2. Nist Net

Nist Net is a free Linux-based package used for network emulation. Much like its descendant NetEm, which was mentioned previously, it runs as an extension of the Linux kernel. It can be used to test and simulate a wide array of WAN properties, such as packet loss, bandwidth limitations, latency, and network congestions. The word "emulation" is defined by the two creators Mark Carson and Darrin Santay as the testing of a network in a simulated environment in addition to a real hardware network setup [4]. The real component in this sense is the actual machine running Nist Net, and the simulated component being the simulation factors such as introducing delays or bandwidth limitations in a logical sense inside of the Nist Net package. Therefore, Nist Net, much like NetEm, benefits from the simulation environment, which is easily changed and reproducible, but also benefits from the factors of a real network device setup [4].

The implementation of Nist Net is simple, as it is implemented and configured through a Linux operating system command line, and, for example, only requires a simple PC-router setup in order to use its emulation functionalities. By changing parameters in the Nist Net configuration, the user can define the desired network manipulation rules in a table of emulator entries, in which the user must also specify for which packets the rules apply, and which emulation factors should be applied [4]. The delay parameter sets the delay of incoming packets in milliseconds and has multiple parameters which can be applied to the delay behavior. For example the added delay can be static, following a random distribution, or by default follow a right tailed delay distribution which closely resembles the actual distribution of ping delays, tested in a three hour connection of machines in a Network, as observed by the authors of Nist Net [4] .

Listing 2: Configuration of Delay in Nist Net

```
# cnistnet -a 0.0.0.0 0.0.0.0 --delay 60
```

The delay parameter in Listing 2 adds 60 ms of simple delay to all traffic passing through a network node running Nist Net [5].

2.3. Dummynet

Dummynet is a network emulation tool developed in the late 1990s [2]. It was originally designed for running configurable experiments in network setups and has been developed for FreeBSD, a Unix-like operating system, for which it later became a default package. Across the years the support for other operating systems was expanded, now supporting Linux, MacOS, and Windows [6]. Dummynet works as a network emulator by utilizing pipes, which are used as a communication link with configurable bandwidth and other factors, such as delay [6]. These pipes are combined with different queuing methods which simulate those used by actual network devices, with FIFO queues being the default setting of Dummynet [6] [2]. The user can choose which traffic gets routed through which pipe, or alternatively, set up a pipe, which accepts any traffic. By defining the parameters of a pipe, for example the delay parameter, any traffic set to pass through the pipe has these parameters applied. The following command snippet shows how to define a simple delay of 60 ms and route all traffic through a pipe, the emulated link [6].

Listing 3: Adding simple delay and configuring traffic to a pipe in Dummynet

```
# ipfw pipe 1 config bw 2Mbit/s delay 60ms
# ipfw add pipe 1 ip from any to any
```

The parameter "bw 2 Mbit/s" in Listing 3 defines the bandwidth and the following "delay 60 ms" sets a delay of 60 ms to pipe 1. The second line routes all the traffic through pipe 1 [6].

Dummynet can be set up as a network router, which accepts the incoming taffic, applies the emulation factors and then forwards it to the network. It can also be established in a distributed fashion, where every device in the network must have Dummynet enabled [2].

2.4. Comparison of NetEm, Nist Net and Dummynet

In some network setups it might be sufficient to test the network behavior under the effects of simple constant latency, but this is sometimes not enough for more complex network setups and testing them under real network conditions. With Dummynet it is only possible to add a constant amount of delay to a given pipe. It does not offer more complex distributions for the latency that are possible in Nist Net and NetEm. When performing simple experiments in a emulated network, Dummynet is definitely sufficient for adding constant latency, but it cannot accurately simulate the latency behaviors of a real network setup with WAN characteristics like it is possible in Nist Net and NetEm [7].

In NetEm and Nist Net, the user can define the delay distribution to be constant, or alternatively specify a more complex distribution so that the latency follows a real latency behavior more accurately. In this regard, Nist Net and NetEm are similiar, allowing for the latency to follow all kinds of distributions with optional correlation [2].

Emulation accuracy is also an important factor to consider, especially for emulating latency behaviors in a network setup. The more accuracy the emulation is able to achieve, the better one can test and approximate the network setup towards the real environment. In the context of the previously mentioned free network emulation tools NetEm, Nist Net and Dummynet, this accuracy is achieved by the used time resolution of the kernel clock [7]. Dummynet is able to utilize the system clock at the maximum of 10 kHz, whereas Nist Net and NetEm under Linux are only able to achieve up to 1 kHz [2]. In more recent Linux kernel versions, NetEm can utilize High Resolution Timers, providing more accurate emulation effects [7].

The point of emulation plays a role in deciding for the optimal setup of a emulation network. Dummynet is able to emulate inbound and outbound traffic, and can be set up as a network router with emulation functionality, or in a distributed way, in which every device in the network will run a copy of Dummynet [2]. NistNet on the other hand is only able to emulate inbound traffic, while NetEm can only emulate outbound traffic [7]. NetEm is effectively setup in a distributed fashion, in which every device has to run NetEm for the emulation network to function properly, whereas Nist Net must be setup as a router between network nodes [2].

It is important to note that Nist Net is currently not in active development. Drawing a comparison for the three tools, Nist Net and NetEm have similar traffic impairment functions, with the possibility of latency distributions. Dummynet on the other hand only offers basic delay and impairment functionalities. Keeping in mind the active development of NetEm and Dummynet combined with their usability, the decision of a free software emulator definitely falls between either Dummynet or NetEm. Including the accurate emulation functionalities and other features useful for network emulation which Dummynet does not have, the choice should fall on NetEm followed closely by Dummynet as the best software emulators.

3. All-in-one Hardware WAN Emulators

While a software emulator setup on a PC could technically also be specified as a hardware emulator, the all-in-one hardware solutions discussed in this paper differ from the commonly used software emulators in regard to their underlying execution platform. While the software emulators running on a PC setup are limited to the resources and computing power provided by the underlying PC, which varies depending on other current system tasks, the all-in-one hardware solutions run on hardware dedicated to the network emulation tasks. Additionally the operating system of the all-in-one solutions are typically optimized towards network emulation as well [8] [9] [2]. The execution of tasks in the all-in-one solutions may also be offered in a Field Programmable Gate Array (FPGA) based environment, which results in even higher execution and emulation speeds of network effects, since the execution of some tasks is directly carried out on the FPGA hardware, instead of relying on higher level software [2] [10].

The specific Hardware Emulators discussed in this paper, however, will be those typically used in a network setup in a corporate environment, which has additional functionalities compared to the available free software emulators. These Hardware Emulators are costly, and, therefore, only a few of them will be used in such environments, in most companies only one of them will be used in order to emulate the whole network [2]. The installation effort of commercial all-in-one hardware emulators compared to the software emulators is much simpler, they need to be physically installed in the testbed setup and can be configured through user friendly GUIs. Additionally, the companies offering these all-in-one solutions often include additional services in case of faulty equipment, or troubles while installing the devices. The software emulators are targeted towards the "do it yourself" networking specialists, and require a certain skill set in networking, such as the Linux traffic control package, and routing the network flow. Additional configuration also needs to be expected when a software emulator is to be used. Where the previously discussed free software emulators and hardware emulators drastically differ, are the speeds at which they support emulation. The software emulators support the speeds of its underlying computers, which will usually remain under 1 Gbit/s, while the Hardware Emulators such as the Netropy 10G or Netropy 100G are built for speeds of up to 10 Gbit/s and 100 Gbit/s respectively, which makes them very suitable for a real network emulation setup [2] [8].

3.1. Apposite Technology Netropy 100G

Netropy 100G made by Apposite Technology allows speeds of up to 100 Gbit/s [8]. The Installation of the Emulator is straight forward. It is installed directly into the testbed, similarly to the installation of a switch or server. Its 100 Gbit/s emulation engine supports 15 concurrently running WAN connections, allowing for complex network setups and concurrent testing [8]. It offers network degradation features similar to the previously discussed free software link emulators, including a variety of bandwidth, latency, latency variation, and packet loss functionalities [8]. The device is able to impose anywhere from 0 to 10 000 ms of delay to each of its simulated WAN links, with the user being able to impose additional distributions similar to NetEm and NistNet, including normal, constant, uniform and many other distributions [8]. The Netropy device lineup also offers features such as a live traffic monitor, recording of loss and delay behaviors and most importantly configuration of the paths and emulation characteristics of the device [8].

3.2. Spirent Attero-100G

The Attero-100G is manufactured by the company Spirent and provides line delay of up to 256 ms for speeds of up to 100 Gbit/s. A delay rate of 256 ms on 100 Gbit/s emulation speeds, translate roughly to 50 000 km of a typical WAN, making it useful for emulating real WAN conditions [9]. It offers similar traffic impairment functions to the Netropy 100G, such as packet corruption, duplication, reordering, bandwidth manipulation, latency and jitter, and does so with a timing accuracy of 5 ns, providing a very swift emulation onto the packets [9]. Through the restful API, the user can reach the web based GUI, and manipulate impairment functions by configuring profiles [9]. The product comes with two profiles as a standard, providing emulation to one incoming and one outgoing stream, but this can be extended to 16 profiles with 8 incoming and 8 outgoing emulated packet flows concurrently [9]. The Attero-100G emulator offers a range of delay functions such as gaussian-, uniform- and not limited to gamma distributions, which can be explained as continuous probability distributions of the latency. The independent traffic flows allow for a different delay set up for each profile, allowing for a complex testing network setup [9].

3.3. Comparison of Hardware Emulators

The Attero-100G and Netropy 100G are both very powerful hardware WAN emulation solutions, and are the golden standard for accurate network emulation. The dedicated hardware, including the emulation engines, allows for emulation at high line rates of up to 100 Gbit/s. The configuration of both devices is as simple as it gets, and they offer an extensive GUI for configuration and monitoring. Because the functionalities of these two devices are so similar, the only decision to be made is how many concurrent connections the emulator should handle. The Netropy 100G is able to support 15 concurrent WAN connections, while the Attero-100G can support 8 connections. The available traffic impairment functions are almost identical, and the user is able to specify custom delay functions via the GUIs. The Attero-100G and Netropy 100G are both powerful solutions and the use of either of these will result in highly realistic emulation environment, perfect for reproducible testing [8] [9].

4. Comparison of Hardware WAN Emulators and Free Software Link Emulators

Having either an all-in-one Hardware WAN emulator or a high-end computer, set up with one of the previously mentioned free software emulators, up and running, the results of latency emulation does not differ significantly between the two. In terms of the ability to emulate a real WAN environment latency behavior, both solutions are able to attain similar results. Taking NetEm and the Netropy 100G for example, the functionalities are very similiar. The delay can be imposed in a simple fashion, but also in a more complex variation distribution. The speeds at which both, the hardware emulators, and software emulators are able to emulate depend on the hardware model, and the software emulators depend on the underlying computer, on which they run on [2]. This certainly also has to be noted, since software emulators realistically cannot reach high emulation speeds of up to 100 Gbit s, and reaching even remotely high speeds of around 10 Gbit s becomes costly, since the underlying PC must be able to provide high computing power for the emulation software. The Netropy 100G and the Attero-100G are specific models that have ports, which support up to 100 Gbit s right from the manufacturer with dedicated hardware and execution platforms [8] [9].

Performance can differ greatly between a software and all-in-one hardware emulator, depending on the hardware used for software emulation. A PC running a software emulator shares its CPU resources with all of the operating system related processes in addition to the emulation software. The CPU resources needed by other processes can vary greatly and in return lead to a lower performance of the software emulator in terms of emulation speeds. This becomes even more apparent when more connections and impairment functions are handled by the software, increasing the CPU load by significant amounts [2]. The hardware emulators on the other hand are specifically built and optimized for handling multiple connections with multiple impairment profiles at the exact speeds which are specified by the different models available [8] [9].

The Commercial all-in-one Hardware WAN Emulators are definitely more suitable for more complex network structures, since the setup, physical installation and configuration is straight forward. The software emulators are more complex to install, since they must be configured manually and the traffic flow through a PC running a software link emulator must be specified as well. The Attero-100G and Netropy 100G offer remotely accessible GUIs with ways to configure network degradation functions and real time performance statistics. This makes the commercial emulators useful for immediately evaluating the network behavior after imposing new delays, packet behaviors, etc. While these factors in terms of user friendliness and network evaluation are a clear improvement to the free software emulators, the high price of the commercial emulators weighs in on a decision between the two.

5. Fiber Optic Delay Lines

Fiber optic delay lines can be considered as another type of Network Emulation Hardware with regard to adding latency. They are another way of introducing fixed delay for network traffic, specifically in a optical network setup using glass fiber cables. A Fiber optic delay line functions by receiving input traffic through a fiber cable connection, then routing the traffic through the length of the spooled fiber cable inside of the device, and routing it back to an output channel [11]. The inherent characteristic of a fiber cable in being relatively resistant to interference factors, and having a low amount of propagation loss, makes this option suitable for adding a low amount of delay in a optical network [11]. Since the amount of delay added is proportionate to the length of the cable inside of the Fiber Optic Delay Line device, the introduced insertion loss will increase, meaning that the strength of light in the input is slowly lost through the length of the cables, but in most cases of hardware available, it is a negligible amount [12] [11]. A downside of using a fiber optic delay line is the increasing size and scale of the device, since the more delay is required, the more wound up fiber cable needs to be present inside of the device [12]. Another downside of this type of hardware are the amount of devices and ports needed to emulate a more complex network, since each fiber optic delay line needs its own switch port [12].

5.1. Fiberplus D8 Series

The Fiberplus D8 is a fiber optic delay line, which is installed in a server rack. The device can hold up to 45 km of fiber, or additionally 90 km of fiber cable spooled up inside of the device [13]. It offers network degradation emulation factors, such as loss, time delay and fiber emulation such as reflectance [13]. The amount of delay added is only available as a constant value, since the delay is not added artificially, but realistically, since the amount of cable that the traffic passes through is actually present. Since the propagation delay of fiber is about the vacuum speed of light [11], the total amount of delay added within the device will be in the example of the Fiberplus D8 in total up to 440 µs with a minimum of 12 µs [13].

6. Conclusion

In conclusion, network emulation is a valuable tool for accurately approximating a test network environment in terms of real network behavior. Accurately simulating the latency behavior of a real network setup composed of multiple network nodes, including WAN structures, is an important step to achieve these real conditions. Multiple approaches were listed and analyzed in this paper. This includes free software link emulators, such as NetEm, Nist Net, and Dummynet, which need an underlying computer to run its emulation software on, but also hardware emulators, such as the all-in-one devices Netropy 100G and the Attero-100G. The use of either hardware all-in-one solutions or software emulators, when setup properly, can achieve similar latency emulation results. Although doing so with high emulation speeds of 10 Gbit s up to 100 Gbit s, all-in-one hardware solutions must be used, as software emulators are limited to their underlying hardware, and realistically cannot reach these high speeds. The configuration efforts and acquisition costs vastly differ between the software and hardware approaches. On the one hand the software emulators discussed in this paper are free to use, they impose a more significant configuration effort and require a certain knowledge of its underlying operating system. On the other hand, the hardware emulators offer an all-in-one package, keeping the installation and configuration simple, and offering user friendly GUIs, with a drawback of high acquisition costs. For a large company, a all-in-one hardware package might be preferable, as one device can be sufficient to emulate a large scale network with high line rates, while the software solutions offer a more budget friendly approach. For fiber optic networks, fiber optic delay lines can provide realistic network emulation, as these devices house the actual length of cable inside, routing the traffic through the actual cable length and not just emulating it within software.

References

[1] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, N. Mckeown, and G. Parulkar, "Can the production network be the testbed?" in *9th USENIX Symposium on Operating Systems Design and Implementation (OSDI 10)*, 2010.

[2] R. Beuran, *Introduction to network emulation.* CRC Press, 2012.

[3] "NetEm," https://wiki.linuxfoundation.org/networking/NetEm, 2021.

[4] M. Carson and D. Santay, "Nist net: a linux-based network emulation tool," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 3, pp. 111–126, 2003.

[5] "NIST Net WAN Emulation Software Installation and Configuration Note," https://www.cisco.com/web/software/280836713/47304/NIST-v4.pdf, 2005.

[6] M. Carbone and L. Rizzo, "Dummynet revisited," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 2, pp. 12–20, 2010.

[7] L. Nussbaum and O. Richard, "A comparative study of network link emulators," in *Communications and Networking Simulation Symposium (CNS'09)*, 2009.

[8] "Netropy Network Emulators," https://www.epsglobal.com/Media-Library/EPSGlobal/Products/files/apposite/N40G-40G.pdf?ext=.pdf, 2017.

[9] "Spirent Attero-100G," https://www.spirent.com/assets/u/spirent_attero-100g_datasheet, 2020.

[10] P. Varga, L. Kovács, T. Tóthfalusi, and P. Orosz, "C-gep: 100 gbit/s capable, fpga-based, reconfigurable networking equipment," in *2015 IEEE 16th International Conference on High Performance Switching and Routing (HPSR)*, 2015, pp. 1–6.

[11] R. Paschotta, "Optical Delay Lines," https://www.rp-photonics.com/optical_delay_lines.html.

[12] T. Zhang, K. Lu, and J. Jue, "Shared fiber delay line buffers in asynchronous optical packet switches," *IEEE journal on Selected Areas in Communications*, vol. 24, no. 4, pp. 118–127, 2006.

[13] "D8 Compact Rack Mount Fiber Optic Delay Line," https://www.gofiberplus.com/Fiber-Optic-Delay-Lines_c_29.html, 2019.

The IEEE 802.11ad Standard: Challenges and Design Adaptations

Mahdi Koubaa, Stephan Günther*, Jonas Andre*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: mahdi.koubaa@tum.de, guenther@tum.de, andre@net.in.tum.de

***Abstract*—After the introduction of IEEE 802.11ad in December 2012, a new multi-gigabit Wi-Fi connection was made possible allowing new applications to be performed. However, using a 60 GHz frequency introduced new problems which were fixed by implementing a new design. This paper describes the faced challenges along with the organization and features of IEEE 802.11ad.**

***Index Terms*—ieee 802.11ad standard, wigig, multi-gigabit wi-fi**

1. Introduction

The 802.11ad standard was introduced primarily to provide a multi-gigabit wireless solution. It uses the 60 GHz carrier frequency and can deliver a data rate of 7 Gbit/s. This standard is mainly used for wireless data transmission and wireless displays.

1.1. The Use of 60 GHz Frequency

The main reason for using a high frequency in 802.11ad is that in higher frequencies a wider bandwidth is achievable without creating interferences in contrast to lower frequencies where interferences are more likely to happen. In the case of 802.11ad, a frequency of 60 GHz allows us to define a single carrier with a bandwidth of 2.16 GHz which is approximately 14 times wider than a single carrier bandwidth in a 5 GHz legacy Wi-Fi frequency [1]. The data rate should increase if we increase the bandwidth while maintaining similar network characteristics, such as the number of users and the modulation and channel encoding scheme. This property of channels has been very useful for building the Multi-Gigabit Wi-Fi.

1.2. Problems Occurring When Using 60 GHz

Due to the small wavelength in higher frequencies, an 802.11ad signal cannot propagate through walls and concrete objects, which means strong signals are originating either from a line-of-sight (LOS) path or from first order reflections on highly reflective materials. Additionally, since the Oxygen absorption of waves peaks at 60 GHz, signal attenuation is strong and signal range will be limited to approximately 10 m [1]. 802.11ad addresses this issue by implementing directional communications through beamforming antenna arrays. These antenna arrays can be weighted to concentrate signal focus in the intended direction and gain wider signal range [1]. This process will be described later and is called beamforming. To facilitate beamforming, the antenna space is partitioned into sectors representing the multiple directions that can be selected.

2. Physical Layer

In a physical layer, data should be encoded by channel encoding to assert a level of transmission failure detection and correction. Additionally, the 802.11ad physical packet is constructed. A few changes have been applied to the structure of the physical packet to adapt to the concept of directional communication. In a later phase, the signal should be modulated on a 60 GHz carrier signal. The IEEE 802.11ad standard supports diverse types of physical layer (PHY) which execute these phases differently.

2.1. Structure of The Physical Packet

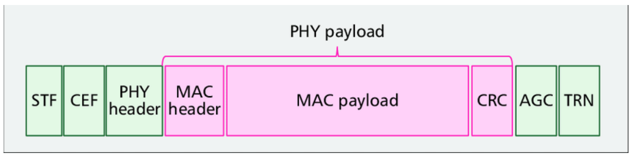


Figure 1: IEEE 802.11ad packet-structure [1]

The IEEE 802.11ad packet begins with a preamble containing a short training field (STF) and a channel estimation field (CEF) which are used in the detection of the implemented PHY and in the reconstruction of the original signal in case of weak channel conditions [2]. The CEF helps the receiver restore the original signal after distortion. The preamble is followed by a PHY Header containing essential information like the modulation and channel encoding scheme (MCS) used to transmit the data. Additionally, it contains the size of the transmitted data. The third part is the data i.e. PHY-Payload generated by the MAC-Layer [2]. An optional automatic gain control field (AGC) follows the data. This field carries information which helps in equalizing the signal amplitude. IEEE 802.11ad introduces a new optional training field (TRN). This field is newly introduced in the 802.11ad standard and is used by stations to train their antennas through beamforming [2].

2.2. Several Types of Physical Layers

To adapt to different use cases, different PHY layers were introduced. The control PHY was intended to work with low signal-to-noise ratio (SNR) operations preceding and during beamforming. Consequently, the control PHY

Packet contains a longer STF field providing a better resistance to unwanted channel effects, compared to the remaining PHY types. Additionally, the control PHY uses MCS0 which implements a BPSK Modulation for better noise resistance and a robust channel encoding scheme with encoding rate $\frac{1}{2}$ to withstand data transmission failures. Since the control PHY is mainly applied to transmit control information between two pairing stations, the packets exchanged contain small data fields with a limit of $1023\,\mathrm{B}$ [2]. Due to the use of a binary modulation scheme, the limited data field size, and the low channel encoding rate, the data rate is limited to $27.5\,\mathrm{Mbit/s}$ when using this PHY Layer.

To achieve higher transmission rates after beamforming, 802.11ad presents the single carrier PHY (SC PHY) and orthogonal frequency division multiplexing PHY (OFDM). Packets in these PHY layers contain data fields reaching $262\,143\,\mathrm{B}$ [2]. In these PHY layers the data rate achieved depends strongly on the MCS used.

In the single carrier PHY (SC PHY) data is modulated on a single carrier signal with a $1760\,\mathrm{MHz}$ bandwidth [2]. When first introduced, this PHY implemented 12 different MCS allowing different data rates scaling from $385\,\mathrm{Mbit/s}$ up to $4620\,\mathrm{Mbit/s}$ [2]. Recently this PHY has been extended with new MCS variants to support $8085\,\mathrm{Mbit/s}$ [2]. To reduce power consumption in mobile devices, a low-power SC PHY was introduced allowing 5 MCS methods. A trade-off is that the data rate in this PHY is limited to $2503\,\mathrm{Mbit/s}$ [2].

The OFDM PHY applies a frequency multiplexing method that modulates data on multiple subcarriers. The low data transmission rates achieved in each subcarrier are added together to result in a high data rate which can reach $6756\,\mathrm{Mbit/s}$ on the complete $1830.47\,\mathrm{MHz}$ band [2]. Using the OFDM PHY results in high energy costs and is no longer the fastest alternative since the extension of the SC PHY. Therefore, this PHY type is obsolete. The 802.11ad hardware is not required to implement all MCS methods which creates differences regarding performance between 802.11ad supporting devices.

3. Personal Basic Service Set

To make use of the directional communication, a new architecture concept "Personal basic Service Set" (PBSS) was introduced allowing peer-to-peer connections between stations [1]. Thanks to directionality, multiple peer-to-peer connections are allowed to coexist without resulting in interferences allowing spatial sharing. However, medium access control (MAC) in a PBSS network must be centralized in one node called "PBSS contol point" (PCP) [1]. The centralization of MAC is necessary for some of the MAC mechanisms described in the upcoming sections. If two stations intend to start a P2P communication in the absence of a PCP, the PCP role must be taken temporarily by one of them. Centralization of PCP can cause the workload to not be distributed equally through the network causing power management issues. Allowing PCPs to hand over the PCP role to other stations would be a good approach to handle this problem [1]. A PCP breakdown causing the whole network to become dysfunctional can potentially represent a vulnerability. This issue can be fixed by implementing an implicit PCP handover procedure which chooses the best alternative PCP when the former PCP is unreachable [1]. PBSS meets the requirements of many applications where ad-hoc-like communications are intended like wireless displays or wireless data storage devices [1].

4. Beacon Interval

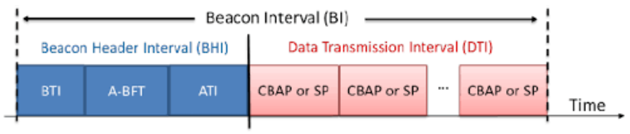


Figure 2: 802.11ad Beacon Interval [1]

The medium access control in a 802.11ad network architecture is managed through periodically recurring beacon intervals (BI) consisting of a beacon header interval (BHI) followed by a data transmission interval (DTI) [1]. The BHI replaces the beacon frame used in legacy Wi-Fi architectures. It includes a beacon transmission interval (BTI). During BTI the PCP/AP performs a sector-level sweeping to transmit beacon frames in all directions. The transmitted beacon frames are used for network announcements as well as for training the PCP/AP transmitter antennas [1]. An explanation of the sector-level sweeping and the beamforming training process is included in the last section of this paper. BTI is followed by an association beamforming training (A-BFT) which is used by stations to train their transmitter antennas and by the PCP/AP to complete its beamforming training. After associating an A-BFT trained station with the PCP/AP, directional communication can be initiated. Since BTI and A-BFT implement the control PHY for a robust association, data rates in these subintervals are too low and result in overheads due to the recurring nature of beacon intervals which constitutes a pertinent problem for some real-time applications such as wireless displays. Solving this problem included the outsourcing of information transmissions from BTI to a new subinterval [1]. As a result, beacon frames transmitted during BTI were restricted to the necessary information to minimize transmission overhead. To reallocate the outsourced transmissions, a new subinterval "announcement transmission interval" (ATI) was defined. During ATI, management information is exchanged with associated stations [1]. This information is necessary for MAC mechanisms used in the DTI. Since associated stations have trained antennas, the control PHY is no longer used in ATI and high data rate transmissions can be performed. The further data transmissions are performed during DTI. The DTI is partitioned into contention-based access periods (CBAP) and scheduled service periods (SP). During a CBAP, stations contend for medium access. The SP is a contention-free period reserved for P2P communication between two assigned stations [1].

5. Medium Access Control

IEEE 802.11ad uses a contention-based mechanism for medium access control. However, the exclusive use of contention for medium access in directional communications can cause problems. To better understand the issues that arise, we first review the contention concept in 802.11ad.

We will then identify the problems. This is followed by an introduction to the adaptation techniques that are in use to address the issues.

5.1. Contention-based Medium Access

Performed in a CBAP, the contention based access in IEEE 802.11ad implements a carrier sense multiple access with collision avoidance (CSMA/CA) expanded with a request-to-send (RTS)/clear-to-send (CTS) exchange. In this medium access method, a node stores for each peer station a network allocation vectors (NAV) timer [1]. A NAV timer is used to know the time left for a peer station in its current communication. NAV timers are usually updated with greater duration field values from RTS/CTS frames received by overhearing communications between other peer stations. When contending for the carrier, a node performs a virtual carrier sensing which is done by evaluating the NAV entry of the destination node. If the NAV entry is non-zero, the next connection attempt will be scheduled after the timer expiration [3]. Otherwise, a physical carrier sensing is executed. The physical carrier sense defines the channel as idle if it does not become busy during a distributed coordination function interframe Space (DIFS) interval. If the channel is sensed idle, an RTS frame is sent to the intended receiver [3]. In case of receiving a CTS frame response, a p2p transmission is initiated. Time elapsing without receiving a CTS or physically sensing an occupied channel is identified as a collision and will cause the next attempt to be scheduled after a *binary exponential backoff* interval. A description of the *binary exponential backoff* can be found in [4].

5.2. Problems Occurring in Contention-Based Access

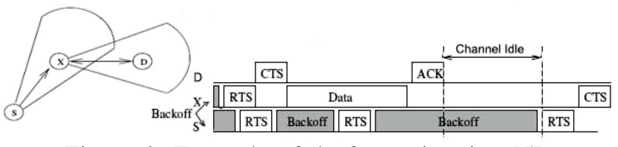


Figure 3: Example of deafness situation [5]

Using contention-based medium access exclusively can be problematic. While waiting for RTS frames, pairing stations do not know the direction of the next transmission. Therefore, nodes are forced to apply a quasi-omnidirectional antenna pattern to deal with such situations, which reduces the receive signal strength. An additional problem is deafness. Fig. 3 shows a situation where a deafness situation arises. We assume the existence of 3 misaligned stations S, D and X. To start a communication with X, S must beamform in the direction of X and contend for the channel. Let us further assume that X already succeeded in initiating a directional connection with D. Since while contending, the RTS/CTS frames are transmitted directionally, such transmissions may not be overheard by S. Consequently, S may not be able to update the NAV timer corresponding to X. As a result, the virtual carrier sensing fails to identify the channel as occupied. If the connection between X and D is long lasting, the station S will experience multiple collisions inducing series of backoff intervals. Due to the nature of binary exponential backoff, there is a high probability for S to be counting down a large backoff interval when X becomes available. Throughout this backoff interval, X might start a new communication with other stations, which could cause more delay for the communication between X and S to take place. This behavior can lead to a severe starvation for some network nodes and create unfairness. While NAV timers cannot always be helpful against deafness, implementing NAV timers remains fundamental to address the deafness problem for contending nodes.

5.3. Hybrid Medium Access Control

To deal with the inefficiencies of the contention-based access, 802.11ad implements a hybrid medium access control combining this scheme with two new mechanisms. The methods in question are Dynamic Channel Time Allocation and Time Division Multiple Access (TDMA).

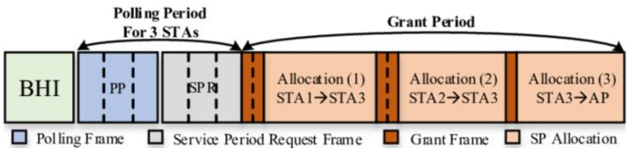


Figure 4: Dynamic Channel Time Allocation [6]

5.3.1. Dynamic Channel Time Allocation. Dynamic Channel Time Allocation is based on a polling phase followed by allocation periods. In the polling phase the PCP/AP sends polling frames to the associated stations. A polling frame schedules the receiving node to submit a service period request (SPR) frame to the PCP/AP to ask for channel time [1]. Afterwards, the PCP/AP allocates channel time according to the SPR frames received. Every allocation period is preceded by an associated grant period. During grant periods grant frames are sent to the individual communication peers of the following allocation period, allowing them channel access. If the PCP/AP takes part in the communication, only one grant frame is transmitted to the non-PCP/AP station [1]. In this medium access approach stations know the directions of the incoming frames since all the instructions come from the PCP/AP. Thus, no quasi-omnidirectional antenna patterns are needed, which improves the signal strength [1]. A Dynamic Channel Time Allocation can be executed in both SP and CBAP periods. When implemented in CBAP, there is a risk that the dynamic channel time allocation becomes disturbed by contending stations that try to acquire the channel. To minimize this risk, the PCP/AP applies a physical carrier sensing method using a point coordination function interframe space (PIFS) interval, which is shorter than the DIFS interval used in contention-based access [1]. This change allows the PCP/AP a more frequent and thus prioritized access to the channel in contrast to other stations. Additionally, extended direction fields are included in the polling and SPR frames to try protecting the polling phase by updating NAV timers of contending stations. The allocation periods are protected by direction fields in the preceding grant frames [1]. If used in CBAP, the PCP/AP can allow contention after performing the requested allocations, otherwise a new polling phase is started [1].

5.3.2. Time Division Multiple Access. Stations using TDMA send resource requests during ATI to request channel time in the upcoming BI.In the next step, the PCP/AP schedules the requested allocations and associates them separately to SPs. The schedule of SPs will then be transmitted by the PCP/AP to all the associated stations in the next ATI [1]. As a result, stations not communicating during an SP enter sleep mode to save energy. This method is oriented to satisfy quality of service (QoS) requirements. Therefore, a resource request should include parameters like the allocation duration and isochronous or asynchronous traffic properties [1]. For duration of communication to be determined, the link in question should completely beam-trained and the transmission rate between the communicating parts should be known [1]. If the traffic stream is isochronous, channel allocations are adapted for meeting certain latency requirements in a constant-rate recurring payload. This type of payload is heavily implemented in wireless displays. In case of asynchronous traffic streams, channel allocations are optimized for non-recurring payload requirements used mostly in file downloads [1]. TDMA makes use of the directional nature of the occurring communications to achieve spatial sharing. Spatial sharing allows noninterfering communications to be initiated concurrently [7].

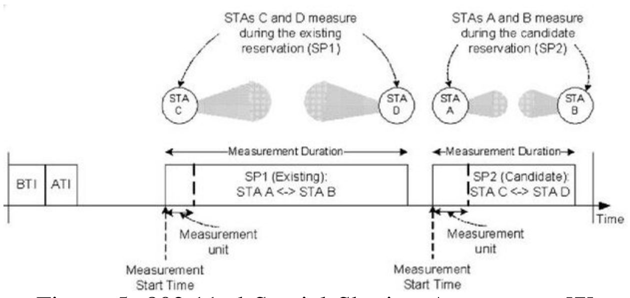

STAs C and D measure during the existing reservation (SP1)
STAs A and B measure during the candidate reservation (SP2)
STA C
STA D
STA A
STA B
Measurement Duration
Measurement Duration
BTI
ATI
SP1 (Existing): STA A <-> STA B
SP2 (Candidate): STA C <-> STA D
Time
Measurement unit
Measurement unit
Measurement Start Time
Measurement Start Time
Figure 5: 802.11ad Spatial Sharing Assessment [7]

Fig. 5 describes how the spatial sharing is performed. We assume the existence of two pairs of communicating peers STA A and STA B, STA C and STA D with corresponding SPs, SP1 and SP2. At first the SPs are allocated by the PCP/AP in different time slots. To test the interference between the two communications, the PCP/AP requests STA C and STA D to beamform and measure average noise plus interference power indicator (ANIPI) during SP1. The same is requested from STA A and STA B during SP2. After completion, the obtained values are transmitted to the PCP/AP. The PCP/AP uses these values to decide if SP1 and SP2 should be allocated concurrently. To prevent interferences in case of sudden changes in the network, the PCP/AP periodically checks the existing spatial sharing configuration by requesting similar measurements to see if the SPs should remain concurrent.

6. Beamforming

To preform beamforming training, two phases are defined: sector-level sweep (SLS) and beam refinement protocol (BRP).

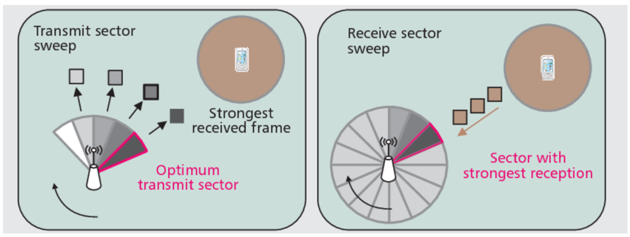


Figure 6: Transmit and Receive Sector Sweep [1]

6.1. Sector-Level Sweep Phase

When performing SLS on a pair of stations, both nodes receive training. The first station to be trained is the initiator, the second is the responder. To train transmitter antennas, a transmit sector sweep (TXSS) is performed. As shown in the left part of Fig. 6, in TXSS the training station sends sector sweep (SSW) frames using different sectors. Meanwhile, the pairing node uses a quasi-omnidirectional receive pattern to measure the SNR values of the received frames. Afterwards, the pairing station reports the optimum SNR and the sector identifier from the corresponding frame in an SSW Feedback [1]. For the receiver antenna training, a receive sector sweep (RXSS) is performed as shown in the right side of Fig. 6. In RXSS, SSW frames are transmitted omni-directionally to the training node which tries to measure the SNR using multiple receive sectors and pick the optimum sector. The SSW feedback for RXSS includes the optimum SNR [1]. During BTI and A-BFT, a sector-level sweep is implemented as follows. A TXSS for the PCP/AP is performed during BTI. Instead of SSW frames, Beacon frames are used to include network announcements.

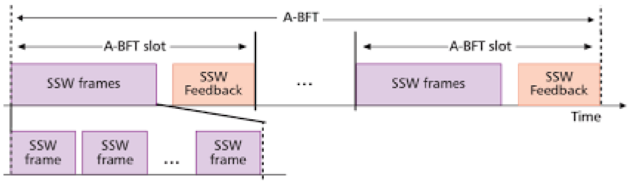


Figure 7: Sector-Level Sweeping in A-BFT [1]

The A-BFT generally includes TXSS for responder nodes. Since multiple responder nodes exist, the PCP/AP prepares multiple A-BFT timeslots. In each A-BFT slot a responder TXSS is performed. The feedback for the PCP TXSS is included in all the SSW frames transmitted during the responder TXSS. Afterwards, SSW feedback for the responder TXSS is sent by the PCP/AP. Stations must contend for accessing a timeslot. Since collisions can be detected if the PCP/AP does not respond with SSW feedback, carrier sensing is not used [1]. The PCP/AP can announce an A-BFT slot for a PCP RXSS [1], which is helpful if a responder node TXSS has already taken place in earlier BIs. Receiver antenna training can also be rescheduled to upcoming SLS or BRP phases in the DTI.

6.2. Beam Refinement Protocol Phase

The BRP procedure tries to refine the antenna directions adjusting the antenna array weights independently from the predefined antenna sectors by trying multiple antenna configurations [1]. This process is used to further improve the signal quality. Since BRP is following an SLS Phase, the pairing node could avoid using the quasi-omnidirectional pattern if a better directional pattern is known [1]. Instead of using multiple frames to test antenna patterns, multiple training fields are included together with parameters like the number of tested patterns inside the same frame. Consequently, using BRP reduces transmission overhead when compared to SLS [1]. This protocol is mainly used in DTI.

6.3. Beam-Training in DTI

As mentioned previously Beamforming training can be used in DTI. When using contention-based access training can be requested directly between stations without the help of a PCP/AP [1]. However, in dynamic channel time allocation and TDMA, training is requested respectively by SPRs and resource requests. The PCP/AP will then transfer the training parameters using grant frames and announcements [1].

7. Linux Support for IEEE 802.11ad

Support for WiGig has been included in the latest Linux kernels through the Wil6210 driver. This driver supports both AP mode, where the node operates as an access point, and station mode. However, the driver only supports Wilocity chips and does not support Intel adapters. In addition, it can be difficult to find a suitable adapter and beamforming antenna array.

8. Multi-Gigabit Throughput with Low Frequency Standards

An example of a multi-gigabit wireless standard that uses low frequency carriers is 802.11ax. This standard uses 1024-QAM modulation on a $160\,\mathrm{MHz}$ bandwidth and a 5GHz or $2.4\,\mathrm{GHz}\,\mathrm{Hz}$ carrier. It can implement $\frac{5}{6}$ rate coding and use frequency multiplexing. In addition, this standard uses Multi-User Multiple-Input Multiple-Output (MU-MIMO) as a spatial multiplexing technique to transmit multiple streams simultaneously using the same carrier frequency. The throughput in this standard can reach $9602\,\mathrm{Mbit/s}$. The use of low carrier frequencies has some advantages, such as long-range signal propagation and backward compatibility with older Wi-Fi standards. However, this approach uses complex modulation schemes that can be sensitive to noise.

9. Conclusion

To adapt to $60\,\mathrm{GHz}$ attenuated signal and LOS propagation, IEEE 802.11ad supports directionality. This paper describes the issues faced and the design adaptations introduced by 802.11ad to integrate this new concept. Different types of PHY are supported; Control, SC, and OFDM PHY, to provide more flexibility towards use cases. The PBSS was intoduced to benefit from antenna directionality and focus on P2P communications allowing new applications e.g. high throughput video streaming. 802.11ad implements a hybrid MAC combining Contention with TDMA and polling methods to overcome the deafness problem. A beamforming training mechanism was implemented to maintain directionality between pairing stations.The 802.11ad standard is partially supported by Linux via the Wil2160 driver. Finally, the IEEE 802.11ad standard has some alternatives in the 5GHz and 2.4GHz wireless standards, such as the 802.11ax standard.

References

[1] T. Nitsche, C. Cordeiro, A. B. Flores, E. W. Knightly, E. Perahia, and J. C. Widmer, "Ieee 802.11ad: directional 60 ghz communication for multi-gigabit-per-second wi-fi [invited paper]," *IEEE Communications Magazine*, vol. 52, no. 12, pp. 132–141, 2014.

[2] B. Schulz, "802.11ad - wlan at 60 ghz a technology introduction," 11 2017.

[3] S. S. N., D. Dash, H. E. Madi, and G. Gopalakrishnan, "Wigig and IEEE 802.11ad - for multi-gigabyte-per-second WPAN and WLAN," *CoRR*, vol. abs/1211.7356, 2012. [Online]. Available: http://arxiv.org/abs/1211.7356

[4] Wikipedia contributors, "Exponential backoff — Wikipedia, the free encyclopedia," 2022, [Online; accessed 10-December-2022]. [Online]. Available: https://en.wikipedia.org/wiki/Exponential_backoff

[5] R. Choudhury and N. Vaidya, "Deafness: a mac problem in ad hoc networks when using directional antennas," in *Proceedings of the 12th IEEE International Conference on Network Protocols, 2004. ICNP 2004.*, 2004, pp. 283–292.

[6] H. Assasa and J. Widmer, "Extending the ieee 802.11ad model: Scheduled access, spatial reuse, clustering, and relaying," 06 2017, pp. 39–46.

[7] "Ieee standard for information technology–telecommunications and information exchange between systems–local and metropolitan area networks–specific requirements-part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications amendment 3: Enhancements for very high throughput in the 60 ghz band," *IEEE Std 802.11ad-2012 (Amendment to IEEE Std 802.11-2012, as amended by IEEE Std 802.11ae-2012 and IEEE Std 802.11aa-2012)*, pp. 1–628, 2012.

Combining Machine Learning With Back-Pressure-Based Routing

Pauline Laßmann, Christoph Schwarzenberg*, Florian Wiedner*
**Chair of Network Architectures and Services, TUM School of Computation, Information, and Technology*
Technical University of Munich, Germany
Email: ga27vit@mytum.de, schwarzenberg@net.in.tum.de, wiedner@net.in.tum.de

Abstract—**The back-pressure routing algorithm guarantees optimal throughput but has poor delay performance. A variety of approaches have been proposed to solve the delay and also memory consumption problems. One way is to use machine learning. The goal of this paper is to find different back-pressure routing policies that are supported by machine learning. Two methods are presented, one using Q-learning and the other using predictive scheduling.**

Index Terms—**back-pressure routing, machine leaning, Q-learning, predictive scheduling**

1. Introduction

Nowadays, applications in areas such as sensor networks, wired flow-based networks and traffic systems require a reliable method to distribute heavy traffic loads across the entire system or network. The back-pressure routing (BP) algorithm offers great potential for such a task. The algorithm examines all possible routes to balance traffic loads across an entire queuing network, thus guaranteeing network-wide throughput optimality [1].

When traffic loads are high, this algorithm works, and available network resources can be used in a highly dynamic manner. However, excessive route searching at low and medium traffic loads can lead to unnecessarily long routes or even routing loops. This leads to poor delay performance [2] [3].

Improvement approaches on various fronts have been made over the years one of them being machine learning aided back-pressure routing. Using prediction its implementations see an overall improvement in delay performance while still being able to efficiently forward packets with near-optimal throughput, having low computational complexity, a distributed implementation and not requiring statistical information about the system dynamics [2] [4].

In the next section of this paper, an overview of different BP algorithms is given. The third section briefly introduces the original BP concept and then presents various framework parameters under which it can be realized. The fourth section deals specifically with BP routing policies supported by machine learning.

2. Related Work

The back-pressure routing algorithm was first introduced 1990 by Tassiulas and Ephremides [1] and initially proposed for wireless multi-hop radio networks. One of its main shortcomings is its poor delay performance.

Over the years there have been a variety of different approaches trying to solve this problem. Each one builds its improvements on a different aspect such as:

- Using shadow queues
- Separating intra-cluster routing from inter-cluster routing
- Using the shortest path algorithm
- Using the last-in-first-out algorithm
- Considering local queue length information
- Eliminating loops in the network
- Introducing a delay parameter

Bui et al. [5] and Athanasopoulou et al. [6] improve the original algorithm with the help of shadow queues. Bui et al. [5] propose shadow queuing as a way of improving delay performance of the original back-pressure algorithm. Athanasopoulou et al. [6] combine the original algorithm with probabilistic routing tables and shadow queues. This way routing and scheduling is decoupled in the network.

In [7] Ryu et al. separate intra-cluster routing from inter-cluster routing. This is done using a two-phase routing method by combining back-pressure routing with source routing. This results in only a subset of nodes having large queues, thus improving delay performance.

The improved algorithms introduced in [8], [9] and [10] make use of the shortest path algorithm to archive better delay performance. Neely et al. [8] introduce BPbias. It combines the information of queue and shortest path length to shorten packet routes. The algorithm of Ying et al. [9], when making each scheduling decision based on the current network load, has a choice between shortest path routing and adaptive routing. The route searching process for the algorithm introduced by Yin et al. [10] dynamically switches between shortest path mode and traditional back-pressure routing mode based on a threshold.

The last-in-first-out algorithm (LIFO) is used in the works of [11] and [12] to improve the original back-pressure algorithm. Moeller et al. [11] combine it with LIFO queuing. Huang et al. [12] prove that near-optimal utility-delay trade-off is achievable with the help of LIFO.

Cui et al. [3] proposed a back-pressure routing algorithm considering local queue length information of up to two-hop nodes and another one considering global queue length information of all nodes, called BPmin.

To eliminate loops in the network Rai et al. [13] propose to use directed acyclic graphs, which in turn improves the delay performance.

The works in [14] and [15] introduce a delay parameter for delay improvement. Ji et al. [15] introduce a back-pressure routing algorithm using a new delay metric to reduce packet delay for light traffic loads. Ji et al. [14] use a new queue management policy with a delay parameter that makes the algorithm select favorable routes by considering both delay requirements and network throughput.
.

Improvement method	Improvement
Using shadow queues	- Improved delay performance - Decoupling routing and scheduling in the network
Separating intra-cluster routing from inter-cluster routing	- Only a subset of nodes have large queues - Improved delay performance
Using the shortest path algorithm	- Shorter packet routes - Improved delay performance - Dynamically switching between shortest path mode and traditional back-pressure routing
Using the last-in-first-out algorithm	- Combining BP routing and LIFO queuing - Near-optimal utility-delay tradeoff - Improved delay performance
Considering local/global queue length information	- Improved delay performance
Eliminating loops in the network	- Improved delay performance
Introduction of a delay parameter	- Reduce packet delay for light traffic loads - Electing favorable routes by considering delay requirements and network throughput - Improved delay performance

TABLE 1: Different back-pressure routing improvement methods and their improvements

3. Back-Pressure Routing Framework Parameters

BP uses time slots to operate. To balance the traffic load in the network, it tries to forward data in each time slot in a way that optimizes the differential backlog between neighboring nodes. This is done by considering all potential routes. In each timeslot, nodes can transmit data that they store in different queues for each destination to a neighboring node. The algorithm forwards packets based on congestion gradients, so it checks which of its neighbours queues for that destination is the smallest and routes the data that way. Data transmitted from one node to another is removed from the first node's queue of the destination and added to the second node's queue of the destination [1]. An example can be seen in figure 1.

As described in section 2, several versions of this algorithm exist. These can have various framework parameters under which they can be realized as seen in Figure 2.

First, back-pressure protocols can be divided into centralized protocols [9], [15] and distributed protocols [4], [2]. It differentiates on where routing and scheduling decisions are made. A coordinator or central server is responsible for routing and scheduling decision making in centralized protocols [9], [15]. High performance can be achieved with routing and scheduling decisions, but on occasion scalability issues due to the high computational complexity can be observed. Distributed protocols [4], [2] are generally more scalable. Network

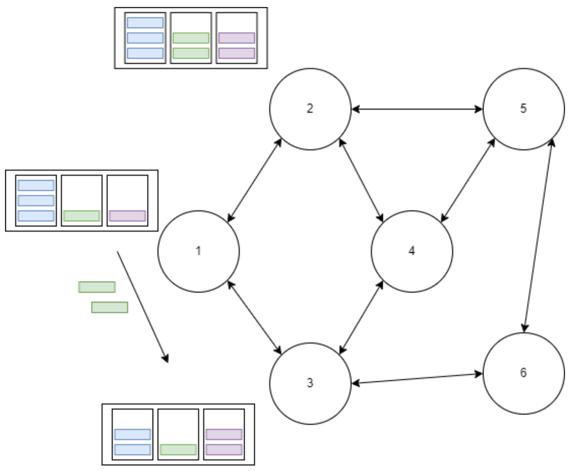


Figure 1: Workings of Back-pressure routing

nodes can use the network state information they maintain to make routing and scheduling decisions. Maintaining the consistency and accuracy of the queue backlog information stored in different network nodes however can be a difficulty. Network performance can be affected by inefficient scheduling and routing decisions when outdated queue backlog information are used [16].

Existing protocols can be classified as adaptive back-pressure routing protocol [11] or fixed back-pressure routing protocol [15]. In adaptive back-pressure routing protocols, the back-pressure scheduling decision based on the queue length primarily determines the next hop of each packet. Fixed back-pressure routing protocols predetermine the route for each flow before the packets are delivered. Back-pressure-based transmission scheduling is used to decide on packet forwarding. However, it has the disadvantage of leading to a minor loss of network capacity [16].

Over the course of time, the original algorithm has been modified again and again in various ways to improve it. Different information of queuing networks such as queue length, path length, clusters and packet delay can be incorporated into the algorithm. Additionally, back-pressure routing in combination with machine learning also gained popularity over the last years [2].

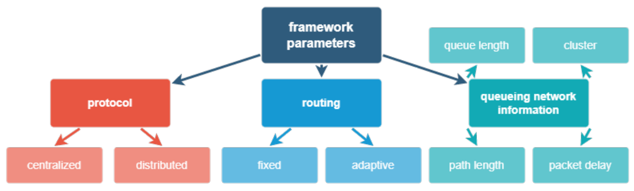


Figure 2: Back-pressure framework parameters overview

4. Back-Pressure Routing in Combination With Data Science

In this section, back-pressure routing algorithms in combination with Data Science are examined in more detail. Q-learning in combination with back-pressure routing is discussed as well as the back-pressure algorithm using predictive scheduling.

4.1. Q-Learning Aided Back-Pressure Routing

Before Q-learning aided back-pressure routing is discussed the concept of Q-learning is conveyed using the Q-routing algorithm. Then the multi-agent Q-learning-based back-pressure routing (QL-BP) algorithm and the adaptive traffic control algorithm are presented.

4.1.1. Q-Routing as a Reinforcement Learning Approach for Packet Routing. Boyan et al. [17] present Q-routing as an algorithm that learns a routing policy which attempts to strike a balance between minimizing the number of hops for packet delivery and the possibility of congestion on popular routes. They refer to their algorithm as a version of the Bellman-Ford shortest path algorithm. For Q-routing to work, a reinforcement learning module is embedded in each node of a network. To keep accurate statistics on which routing decisions result in minimal delivery times, only local communications are used. Furthermore, the Q-routing algorithm is able to route efficiently even when critical aspects of the simulation, such as network utilization, are allowed to vary dynamically.

Based on experiments with different routing policies, the algorithm selects the one to use. Reinforcement learning can be used to update the selected routing policy faster. The performance of a policy is measured by the total time it takes to deliver a packet. To calculate this, Q-learning uses a "learning rate" parameter, as well as an old time estimate and a revised time estimate for packet delivery, to obtain a solution [17].

Q-learning has the disadvantage of being greedy and therefore cannot fine-tune a shortcut discovery strategy. One solution presented in the paper is for the algorithm to select routing directions with a degree of randomness in the initial learning phase. Since this would have an extremely negative impact on congestion, a node uses what is called a "full echo" modification instead of sending actual packets in a random direction. Using this, a node sends information requests to its immediate neighbors each time it needs to make a decision. Each neighbor sends back an estimate of the total time to reach the destination. If shortcuts appear or the policy is inefficient, this information quickly propagates through the network and the strategy is adjusted accordingly. This revised Q-routing is referred to as "full-echo" Q-learning [17].

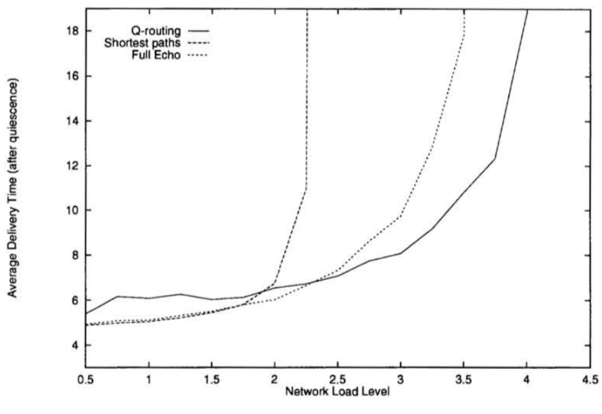


Figure 3: Delivery time for Q-routing, "full echo" Q-routing and shortest path routing [17]

As seen in Figure 3 Q-learning exhibits initial inefficiency when traffic load is low compared to the shortest-path routing strategy, because it first learns the network topology. Once the learning phase is overcome, it performs equivalently to the shortest path. Q-routing with "full echo" is indistinguishable from the shortest path strategy. As the network load increases, the shortest path routing strategy is outperformed by Q-routing with "full echo". Q-routing performs best because it learns an efficient routing strategy and continues to route that way. Q-routing with "full echo", on the other hand, constantly changes its strategy under high load. Not until a further significant increase in traffic load does the Q-routing algorithm also succumbs to overload [17].

4.1.2. Multi-agent Q-learning-based back-pressure routing (QL-BP) algorithm. Gao et al. [2] propose the multi-agent Q-learning-based back-pressure routing (QL-BP) algorithm. They take a general delay reduction framework based on information of the queuing network (bias) and build their QL-BP algorithm on it. The framework goes through three stages:

- Information collection: in this stage useful, local or global, information is collected including queue length, shortest path and packet delay
- Bias extraction: in this stage useful features (such as route congestion estimation) are extracted either in a heuristic manner or with the aid of machine learning based methods like Q-learning
- Back-pressure routing: the extracted bias are programmed into the back-pressure routing algorithm after which the algorithm is capable of adaptively changing packet routes

Each node maintains multiple Q-learning agents that are responsible for generating route congestion estimates from the collected information in the bias extraction phase. Each agent updates the route congestion estimate using the queue length information and the route congestion estimates of the neighboring nodes. Since route congestion is estimated using only local information from neighboring nodes a distributed implementation is possible. Based on the estimated route congestion, each node routes packets to their destinations along the least congested routes [2].

The QL-BP algorithm can be further improved by considering information about the shortest path (QLSP-BP). In this case, the QL-BP algorithm remains the same, except that the shortest path between a source and a destination node is considered in the bias extraction [2].

The QL-BP algorithm is able to maintain a distributed implementation, low computational complexity, and an optimal throughput rate. It reduces the average packet delay by 71% compared to the original BP algorithm at low traffic load. At moderate traffic load, it is 82% higher. The QL-BP algorithm effectively learns the congestion of the routes and adaptively reroutes the packets to better routes. For this, a slight amount of packet delay is accepted in favor of distributed algorithm implementation and low computational complexity. As mentioned earlier, the QL-BP algorithm can be significantly improved by considering shortest path information. The QLSP-BP algorithm outperforms all variants of back-pressure routing algorithms. It reduces the average packet delay by 95%

under light traffic load and 41% under medium traffic load and is the best variant of the improved back-pressure routing algorithms [2]. Figure 4 shows all this graphically.

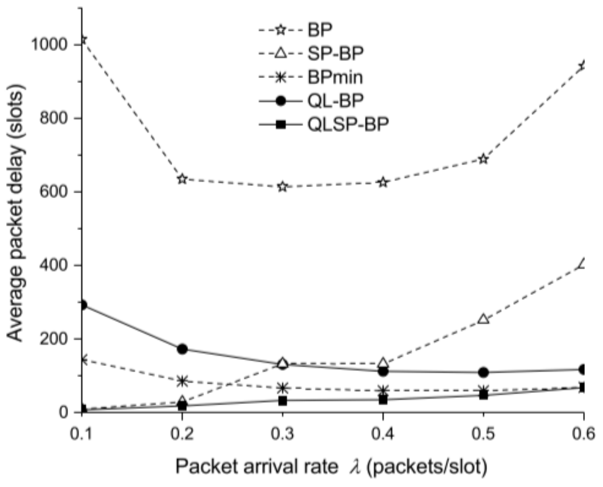


Figure 4: Packet delay for different back-pressure algorithms [2]

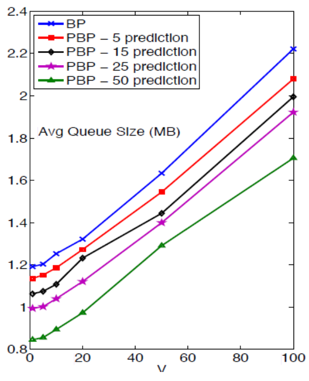
4.1.3. Adaptive traffic control algorithm. Maipradit et al. [18] also use the Q-learning-based back-pressure algorithm. They use it as an adaptive traffic control algorithm. They manage to significantly decrease the average vehicle travel time from 16% to 36% compared to other algorithms. Although this algorithm is applied for traffic control, it should be easily transferable to routing networks. Each intersection has a control agent. This agent collects vehicle speed and vehicle position information in each time window. Congestion information is also exchanged between neighboring agents. Based on the exchanged congestion information, the agent updates its own congestion estimate based on Q-learning. Eventually, all agents receive global congestion information. These are helpful in tasks two and three of the three tasks that each agent performs in every time slot: Learning global congestion information, selecting the optimal traffic phase based on the back-pressure algorithm and vehicle steering, where after a vehicle passes the intersection and enters the next road under the traffic phase selected in task two, the agent determines which lane of this road the vehicle shall use [18].

Their adaptive traffic control algorithm based on back-pressure and Q-learning (ARD-BP-Q) is decentralized and the agent at each intersection executes the algorithm independently. An additional feature is that vehicles with longer travel times pass through an intersection first [18].

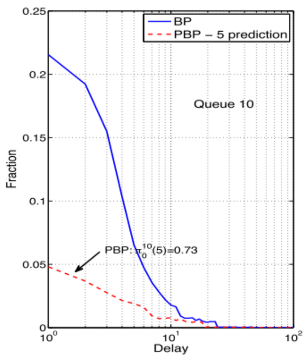
4.2. Predictive Scheduling Aided Back-Pressure Routing

Huang et al. [4], discuss predictive scheduling. Using a look-ahead window model for pre-allocating rates the delay performance of the original back-pressure algorithm is improved. They draw inspiration from pre-fetching techniques used in memory management, branch prediction in computer architecture, and current advances in data mining for learning user behavior patterns. The model is implemented using prediction queues created by the server based on the previous packets.

The authors propose the predictive back-pressure (PBP) algorithm, which performs the BP algorithm based on the prediction queues. PBP achieves a cost performance that is arbitrarily close to optimality. At the same time, it guarantees that the average system delay vanishes as the size of the prediction window increases. Moreover, PBP retains all the desired properties of the original back-pressure algorithm. It remains greedy and does not require statistical information about the system dynamics. In addition, the look-ahead window helps the server use connections more efficiently. The queuing policy chosen for the look-ahead queue leads to different improvements. With first-in-first-out (FIFO) queuing, PBP achieves an average reduction in packet delay that is linear with the size of the prediction window as seen in Figure 5. With last-in-first-out (LIFO) queuing, the average packet delay decreases exponentially with the window size as seen in Figure 5. Thus, the average delay under PBP is strictly better than under the original back-pressure algorithm and totally vanishes as the prediction window size increases [4]. The authors of [4] prove that the algorithm achieves a cost performance arbitrarily close to optimality and that the prediction is more accurate with a larger window size.



Average queue seize of different prediction windows using PBP with first-in-first-out queuing policy (V being a control parameter used to tradeoff utility performance and system delay) [4]



Packet delay distribution using PBP with last-in-first-out queuing policy [4]

Figure 5: PBP performance results

5. Summary and Conclusion

In this paper, the original back-pressure routing algorithms, ones using machine learning and one making use of other improvement methods were presented.

The goal of this work was to find several back-pressure routing policies supported by machine learning, about which Table 2 gives an overview. .

The multi-agent Q-learning aided back-pressure routing algorithm [2] is able to significantly improve delay performance and maintain the following attractive features: distributed implementation, low computational

Variant	Improvement
Multi-agent Q-learning aided back-pressure routing algorithm	- Improved delay performance - Distributed implementation - Low computational complexity - Throughput optimality
Adaptive Traffic Control Algorithm	- Reduce average travel time - Decentralized - Longest travel times passes first
Predictive Back-Pressure algorithm	- Better cost performance - System delay vanishes with increasing prediction window size - No statistical information about system dynamics required - Greedy

TABLE 2: Comparison of machine learning aided back-pressure routing algorithms

complexity, and throughput optimality. A similar system is also used for the adaptive traffic control [18] algorithm, where the traffic delay is also significantly reduced.

The PBP (predictive back-pressure) [4] algorithm, based on a lookahead prediction window model, achieves cost performance arbitrarily close to the optimum. At the same time, it guarantees that the average system delay vanishes as the size of the prediction window increases.

We found that at this stage, only two back-pressure algorithms supported by machine learning could be found. Q-routing and predictive scheduling. Since both work in their specific theoretical models introduced in the respective paper it is however difficult to compare them in effectiveness and suitability to other network models. They are nevertheless both able to significantly reduce delay power and efficiently forward packets with near-optimal throughput, but face other challenges. More computation time is required, and nodes must constantly record data and update the parameters stored there. Since both discussed algorithms based on machine learning have advantages over other state-of-the-art back-pressure algorithms, especially in throughput optimality and cost performance, we believe that both Q-learning and predictive scheduling are attractive optimizations of the original algorithm.

The multi-agent Q-learning based back-pressure routing algorithm has already been improved using the shortest path algorithm [2]. Even though the presented machine learning based back-pressure routing algorithms are already a major improvement over the original, this proves there is still room for further optimization to be found in the future.

References

[1] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," in *29th IEEE Conference on Decision and Control*, 1990, pp. 2130–2132 vol.4.

[2] J. Gao, Y. Shen, M. Ito, and N. Shiratori, "Multi-agent q-learning aided backpressure routing algorithm for delay reduction," 2017. [Online]. Available: https://arxiv.org/abs/1708.06926

[3] Y. Cui, E. M. Yeh, and R. Liu, "Enhancing the delay performance of dynamic backpressure algorithms," *IEEE/ACM Transactions on Networking*, vol. 24, no. 2, pp. 954–967, 2016.

[4] L. Huang, S. Zhang, M. Chen, and X. Liu, "When backpressure meets predictive scheduling," *IEEE/ACM Transactions on Networking*, vol. 24, no. 4, pp. 2237–2250, 2016.

[5] L. Bui, R. Srikant, and A. Stolyar, "Novel architectures and algorithms for delay reduction in back-pressure scheduling and routing," in *IEEE INFOCOM 2009*, 2009, pp. 2936–2940.

[6] E. Athanasopoulou, L. X. Bui, T. Ji, R. Srikant, and A. Stolyar, "Back-pressure-based packet-by-packet adaptive routing in communication networks," *IEEE/ACM Transactions on Networking*, vol. 21, no. 1, pp. 244–257, 2013.

[7] J. Ryu, L. Ying, and S. Shakkottai, "Back-pressure routing for intermittently connected networks," in *2010 Proceedings IEEE INFOCOM*, 2010, pp. 1–5.

[8] M. Neely, E. Modiano, and C. Rohrs, "Dynamic power allocation and routing for time varying wireless networks," in *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, vol. 1, 2003, pp. 745–755 vol.1.

[9] L. Ying, S. Shakkottai, A. Reddy, and S. Liu, "On combining shortest-path and back-pressure routing over multihop wireless networks," *IEEE/ACM Transactions on Networking*, vol. 19, no. 3, pp. 841–854, 2011.

[10] P. Yin, S. Yang, J. Xu, J. Dai, and B. Lin, "Improving backpressure-based adaptive routing via incremental expansion of routing choices," in *2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, 2017, pp. 1–12.

[11] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali, "Routing without routes: The backpressure collection protocol," in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, ser. IPSN '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 279–290. [Online]. Available: https://doi.org/10.1145/1791212.1791246

[12] L. Huang, S. Moeller, M. J. Neely, and B. Krishnamachari, "Lifo-backpressure achieves near-optimal utility-delay tradeoff," *IEEE/ACM Transactions on Networking*, vol. 21, no. 3, pp. 831–844, 2013.

[13] A. Rai, C.-p. Li, G. Paschos, and E. Modiano, "Loop-free backpressure routing using link-reversal algorithms," in *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. MobiHoc '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 87–96. [Online]. Available: https://doi.org/10.1145/2746285.2746304

[14] Z. Ji, Y. Wang, and J. Lu, "Distributed delay-aware resource control and scheduling in multihop wireless networks," in *2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall)*, 2015, pp. 1–5.

[15] B. Ji, C. Joo, and N. B. Shroff, "Delay-based back-pressure scheduling in multihop wireless networks," *IEEE/ACM Transactions on Networking*, vol. 21, no. 5, pp. 1539–1552, 2013.

[16] Z. Jiao, B. Zhang, C. Li, and H. T. Mouftah, "Backpressure-based routing and scheduling protocols for wireless multihop networks: A survey," *IEEE Wireless Communications*, vol. 23, no. 1, pp. 102–110, 2016.

[17] J. Boyan and M. Littman, "Packet routing in dynamically changing networks: A reinforcement learning approach," in *Advances in Neural Information Processing Systems*, J. Cowan, G. Tesauro, and J. Alspector, Eds., vol. 6. Morgan-Kaufmann, 1993. [Online]. Available: https://proceedings.neurips.cc/paper/1993/file/4ea06fbc83cdd0a06020c35d50e1e89a-Paper.pdf

[18] A. Maipradit, J. Gao, T. Kawakami, and M. Ito, "Adaptive traffic control algorithm based on back-pressure and q-learning," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 1995–1999.

Mechanisms and Protocols for Reliable Communication Networks

Désirée Rentz, Henning Stubbe*, Kilian Holzinger*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: desiree.rentz@tum.de, stubbe@net.in.tum.de, holzinger@net.in.tum.de

Abstract—**When creating communication networks, technical problems occur that can disrupt or even break the communication. To overcome the technical challenge of making such communication reliable, various mechanisms have been invented. After mentioning some common failures that negatively affect the functionality of a network, this paper summarizes a selection of mechanisms that have been created to improve the reliability of communication networks, and explains some metrics that can be used to characterize and compare these mechanisms.**

Index Terms—**reliability, time sensitive networking, interference, congestion**

1. Introduction

There are many different examples for communication networks, such as ACARS, which is used in aviation, or DVB-S2, a standard for satellite television broadcast. This paper discusses the reliability mechanisms used for the Internet. However, their application is not restricted to that; for example the error correcting code briefly described in Section 3.1 is also used in DVB-S2. [1], [2]

1.1. Definition of Reliability

Since there is no generally applicable definition of reliability, we will first examine some aspects contributing to it.

From a more abstract perspective, Zhang et al. divide the analysis of network reliability into three layers: connectivity, performance and application reliability. The first refers to network topology and physical connectivity, the second is described as the "probability that performance indicators remain their values within expected ranges under a certain traffic flow" [3]. In their work on network reliability testing, Li et al. describe network reliability as the ability to ensure the functionality of the network. They summarize this as "transmitting data timely, completely and correctly", which are measurable quantities [4]. In another paper dealing with wireless network routing, Biswas et al. describe reliability as "a mission-specific metric evaluating the probability that a packet gets delivered with a given deadline". They consider it "to be a combination of availability and dependability" [5]. Similarly, Shi et al. consider the "reliability of a network defined as the probability of successful communication" [6].

Based on these statements, we can conclude that in order for a communication network to be considered reliable, it must offer high availability (i.e. data transmission to the destination must be possible at any given point in time), and correct, complete and timely transmission must be ensured. The degree to which the latter three requirements must be fulfilled depends on the application.

To avoid ambiguity, it should be noted that this does not include performance increases beyond the absolute minimum that is defined by the application. The remainder of this paper also assumes that there are no intentional attacks on the reliability of the network.

1.2. Types of Failures

The first step of improving the reliability is to analyze the typical impediments during data transmission. Shi et al. summarize that "the failures of computer networks usually consist of two modes: connective failures and congestion failures" [6].

Due to the limitations of the network participants, too much traffic can lead to congestion failures. These can manifest in the form of lost or delayed packets or failing to establish new connections in connection-oriented protocols.

Connectivity failures are failures in which the transmission of data between two nodes is compromised. The data arriving at the receiving end may contain bit errors of which the receiver is unaware. This can range from a single flipped bit to completely corrupted data. When the quality of a link has degraded to the point where data can no longer be transmitted, this is referred to as a link failure.

2. Background

In preparation for the next section, this section will first provide a general overview of the relevant protocols and provide some background knowledge.

The inner working of the Internet is realized through a variety of protocols. The protocols used in Internet communication are usually categorized according to the OSI model, in which each protocol of an upper layer builds on the layers below it, with the lowest layer being the physical communication. The layer above does not necessarily need to know about the implementation of the layer below.

Figure 1 contains the protocols that are used as examples in the following. The model provides a rough overview, but protocols cannot always be clearly assigned to a layer. Some protocol specifications may span multiple layers, as it is the case with Ethernet. Other protocols work within a layer, together with another protocol. For

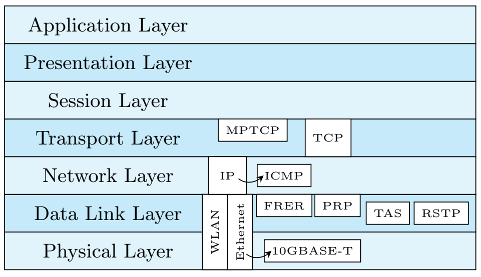


Figure 1: OSI model with exemplified protocols

example, MPTCP can optionally be stacked onto TCP, and still operate within the transport layer.

Ethernet (IEEE 802.3). Ethernet is a protocol standardized by the IEEE 802.3 working group. Its specification covers both the physical and data link layer and is used for Local Area Networks and Metropolean Area Networks (LAN/MAN). The physical layer uses wired connections, and offers multiple implementations for copper and fiber wiring using both full-duplex and half-duplex configurations. For example, 10GBASE-T Ethernet is based on a full-duplex copper medium, and supports a data rate of 10 Gbit/s. [7, Clause 1 and 55]

WLAN (IEEE 802.11). The IEEE 802.11 working group creates standards to implement WLANs (Wireless Local Area Networks). The WLAN specification covers both the physical and data link layer. As with Ethernet, several options are defined for the physical layer, using different frequency bands. Wireless communication is half-duplex and generally more susceptible to bit errors compared to wired media. [8, Clause 4 and 8]

Time Sensitive Networking (IEEE 802.1 TSN). One part of the IEEE 802.1 working group is the Time Sensitive Networking (TSN) task group. It develops a collection of standards aimed at "providing deterministic connectivity through IEEE 802 networks." Two standards we will look briefly at are IEEE 802.1CB (Frame Replication and Elimination for Reliability, FRER) and IEEE 802.1Qbv (Time Aware Shaper, TAS). [9]

Transmission Control Protocol (TCP). The Transmission Control Protocol (TCP) was developed together with the Internet Protocol (IP). It enables stream-oriented data transmission using connections. Since IP explicitly does not implement reliability mechanisms, TCP makes very few assumptions about the reliability of the underlying layers. [10], [11]

3. Mechanisms to Improve Reliability

In the following, we will look at certain types of failures and the corresponding mechanisms.

3.1. Dealing with Interference

Electronic interference can be caused by external sources, but also by the electronic equipment itself. It usually leads to bit errors. A special type of interference is crosstalk, where two parallel data lines influence each other due to electromagnetism.

Physical Wiring. A very common strategy to minimize the crosstalk between two parallel pairs of wires within a cable is to twist the wires inside around each other. Additionally, the interference from external sources can be reduced by shielding the cable, using a woven copper shield, or wrapping the cable in foil. These cables are commonly referred to as Unshielded Twisted Pair (UTP) and Shielded Twisted Pair (STP) cable depending on whether shielding was used or not. In buildings, the physical placement of cables and wireless stations should also be considered, especially when sources of high interference exist. [12]

Signal Processing. If the interference is known to the transmitter, there are methods for the transmitter to precode a signal before sending. One type of precoding is known as Dirty Paper Coding (DPC). The fundamental idea is to modify the sent signal in such way that if the interference is added, the receiver will be unaware of the interference. One such DPC is the Tomlinson-Harashima precoder, which is used in 10GBASE-T Ethernet to cancel out near-end crosstalk, a type of crosstalk detected on the same side where the signal was sent. [12], [13], [7, Clause 55]

Error Correcting Codes. By encoding the digital data with an error correcting code before transmission, some bit error patterns can be corrected by the receiver. One example of an error correcting code is low density parity control (LDPC), which used in 10GBASE-T Ethernet. It calculates parity bits using a sparse matrix, where each data bit is included in at least two parity bits. [7, Clause 55], [14]

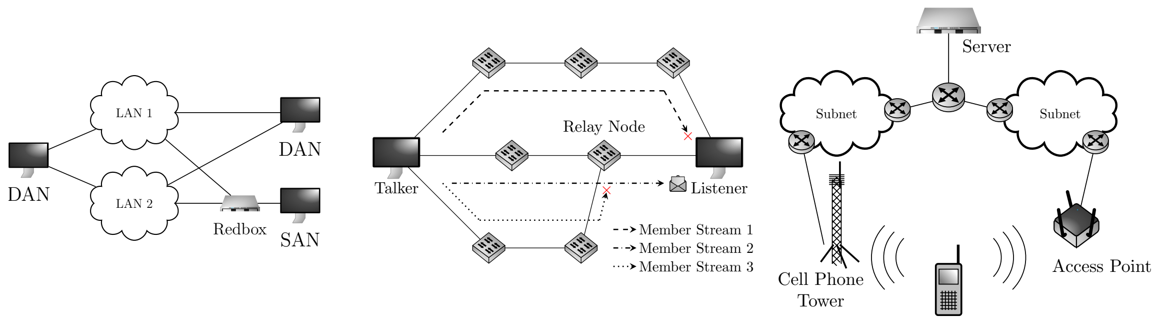
Error Detection Codes. In contrast to error correcting codes, error detection codes aim to detect as many error patterns as possible, but without being able to reconstruct the original data. One such code is the Cyclic Redundancy Check (CRC), which is based on polynomial division. At the data link layer, a 32-bit wide CRC code is used in the Frame Check Sequence field of the Ethernet and the WLAN header. [7, Clause 3], [8, Clause 9]

Additional Checksums. Several network and transport layer protocols implement an additional checksum value to verify the correctness of the received data. TCP, for example, calculates a checksum from a 16-bit wide sum of the data using one's complements. Other examples of third and fourth layer protocols that use checksums include UDP, IPv4 and ICMP. The usefulness of additional checksums has been confirmed in literature. [10], [11], [15]–[17]

3.2. Dealing with Collisions

If two nodes send data at the same time in half-duplex configurations, the signals overlap and become unusable. This is commonly referred to as collision.

CSMA/CD in Half-Duplex Ethernet. Ethernet implements a mechanism called CSMA/CD that aims to avoid collisions in half-duplex setups. The algorithm can be thought of as multiple people (Multiple Access) talking



(a) Parallel Redundancy Protocol (b) Frame Elimination and Replication (c) Multipath TCP

Figure 2: Redundant path usage with different protocols

at a table: Whenever a person has something to say, they check whether anyone else is speaking, before they talk (Carrier Sense). If two people start talking at the same time, they stop, wait for a randomly chosen time interval, and then try again (Collision Detection). [18]

However, today's networking devices most commonly use the full-duplex mode. Full-duplex does not require the use of CSMA/CD, since collisions cannot occur.

CSMA/CA in WLAN. While collisions can be fully avoided in Ethernet using full-duplex connections, a wireless network is always half-duplex. Unlike wired connections, the medium cannot be listened during transmission. Therefore, CSMA/CD cannot be directly applied to WLAN.

As with CSMA/CD, the medium is checked before transmission is attempted. If the channel is free, a distributed coordination function determines a time interval that the endpoint waits before attempting to send, to give other members in the channel a chance (Collision Avoidance). The waiting time consists of a randomized time interval and a constant interframe space.

The proper detection of whether the channel is clear may not be possible if a transmitting station is out of range of the station initiating the transmission. Both endpoints may be within range of the WLAN base station but not within range of each other. This is commonly referred to as the hidden station problem. To avoid collisions caused by this, an additional method can be implemented in which an endpoint in the network sends a request-to-send (RTS) frame to which the base station will reply with a clear-to-send (CTS) frame when permission is granted to transmit data. [19]

3.3. Dealing with Link Failures

When a link in the network becomes unusable, it is called a link failure. The consequence of this is a changed topology, which in turn changes the routing decisions that the nodes on the path must make. Usually, nodes communicate to each other about link failures using routing protocols.

Rapid Spanning Tree Protocol. The Rapid Spanning Tree Protocol (RSTP) is defined in IEEE 802.1w. Its purpose is to create a spanning tree from a network, i.e. to reduce the network topology to a tree so that any two nodes are connected by a single path only. In the event of a link failure, it adapts to the changed topology. Compared to its predecessor, the Spanning Tree Protocol (STP), RSTP is able to react much faster to topology changes. [20]

Parallel Redundancy Protocol. The Parallel Redundancy Protocol (PRP) is defined as part of an IEC norm. It is intended to guarantee no latency in the event of a link failure by sending each frame over multiple paths. PRP builds on Ethernet and identifies related frames using an additional header attached to the PDU.

PRP imposes requirements on the network topology, as shown in Figure 2a. Each node in the network should be connected to two separate LANs. These Doubly Attached Nodes (DANs) require hardware support. If a node does not support this, a Single Attached Node (SAN) can be placed behind a redundancy box (Redbox) to achieve the dual connectivity. The Doubly Attached Nodes and Redboxes are capable of sending and receiving duplicate frames, and eliminating them if necessary. In the event of a single network link failure, the frame arrives at its destination without any latency, as it would be the case with RSTP, because of these hardware requirements. [21]

Frame Replication and Elimination. Frame Replication and Elimination (FRER) is part of the IEEE 802.1 TSN specifications. Similar to PRP, it is designed to send frames over multiple paths (replication) and to provide a mechanism to eliminate duplicates at the receiving end (elimination). Again, the goal is to avoid any latency in the event of a link failure. FRER does not impose any topology requirements, but redundant hardware paths are required for it to have any effect.

FRER calls a series of frames from a talker to one or more listeners a component stream. A component stream is split into multiple member streams, with each member stream sent over a different path by duplicating the frames. When a relay or listener node receives duplicate frames, it will eliminate one of them, as shown in Figure 2b. FRER identifies the member streams by a so-called redundancy tag or by using a PRP header. In this regard, PRP and FRER are partially compatible with each other. [22], [23]

Multipath TCP. Consider the use case of a person walking from one building to another with a cell phone. As shown in Figure 2c, a cell phone has two interfaces through which it can connect to the Internet, either through a WLAN access point or through a cell phone tower. Depending on the location of the person holding the device, one signal may be stronger than the other.

With connection-oriented protocols, it is not as easy to switch to a different interface because TCP is not designed to handle changing IP addresses. This is where Multipath TCP (MPTCP), an extension of TCP, comes into play: It enables the use of redundant paths by using two or more interfaces of a device for a single connection, which may have different IP addresses. MPTCP keeps track of IP addresses using subflows. It introduces additional header fields that are prepended to the transport layer PDU. [24]

3.4. Retransmissions

Despite the possibility to repair or reroute data, sometimes data is simply lost irretrievably. In this case, the data should be retransmitted automatically. The methods for doing this are usually called Automatic Repeat Requests (ARQ) and use acknowledgement signals to confirm a successful transmission. For efficiency reasons, several data packets are usually sent at once. Sliding window algorithms such as Go-Back-N or Selective Repeat are used to keep track of unconfirmed data. They use a send window and a receive window to regulate the amount of data that is sent “at once”.

Retransmissions in WLAN. After a transmission with CSMA/CA, the transmitter alone cannot determine whether a collision has occurred or not. After sending the data, the sender waits for an ACK control frame from the access point. If the ACK control frame is not received, the sender retransmits the frame. A sender tries a configurable number of times before determining that communication has failed. For example, the implementation described in reference [25] does four or seven attempts, depending on the size of the PDU. This mechanism aims to create conditions similar to wired connections for the upper layers, since bit error are much more likely in the wireless medium. [8, Clause 4]

Retransmissions in TCP. With TCP, each byte sent between two nodes is acknowledged by the receiver using sequence numbers. The initial sequence numbers are exchanged during the three-way handshake. By observing the absence of acknowledgement frames, the sender can retransmit the data until the transmission was successful. The completeness of the transmission can be ensured by closing the connection, signaling the communication partner that the transmission is done. [11]

3.5. Dealing with Congestion

Congestion occurs when nodes on the network reach their limited capabilities, usually processing speed or buffer size. For example, a server in a data center may process data much faster than a mobile device, or a router in the network may become overwhelmed if it has to buffer too many packets at once, e.g. if data needs to be forwarded from a faster link to a slower link.

TCP Flow Control. The TCP flow control mechanism avoids congestion at the receiver. The receiver can announce the number of bytes it is willing to receive by using the receive window field in the TCP header. The sender then configures its sliding window procedure so that it does not send more than this number of bytes at once.

TCP Congestion Control. To avoid congestion in the network, TCP slowly increases the send window until it reaches the capacity that the network can currently handle. To test the capacity of the path, the transmit window is initially increased exponentially ("slow start"). After reaching the limit, which is usually determined by receiving multiple ACK signals acknowledging the same bytes, it halves the window and enters the "congestion avoidance" phase, where it approaches the capacity linearly.

Time Aware Shaping. TCP congestion control can prevent data loss due to congestion, but it cannot guarantee a maximum transmission time. Time Aware Shaping (TAS) addresses this problem by defining a different approach to media access at the data link layer by dividing the time on the channel into cycles, which in turn are divided into time slices. It introduces priorities for the frames and reserves one time slice for each priority within each cycle. In this way, if each hop to a destination keeps the link free for prioritized traffic, a maximum latency can be enforced as long as the prioritized traffic does not reach the limit of its own time slice. [26]

3.6. False Friends

It should be noted that some protocols implement mechanisms that at first glance are related to failures, but do not actively improve reliability. One such example is ICMP, which is part of IP and transmits error messages when an IP packet could not be delivered. IP itself is not a reliable protocol. The documentation for ICMP states, “The purpose of these control messages is to provide feedback about problems in the communications environment, not to make IP reliable.” [15]

4. Analysis

Table 1 gives an overview of which mechanism addresses which failure, and on which layer the corresponding protocol operates. From this we can now draw a few conclusions.

4.1. Evaluation

First of all, reliability problems are mostly due to hardware limitations. While some of these problems can be addressed directly, e.g. by improving the signal or increasing buffer sizes of the nodes, there is only so much that can be done on the physical layer. The advantage of the Internet’s multi-layered design is that even if these hardware-related errors cannot be fixed, the layers above them (primarily the data link and transport layer) can mitigate them. For this purpose, the protocols at the higher layers can implement one or more reliability mechanisms,

Mechanism	Failure	Layer	Technique	Improved Reliability Aspect
Physical Wiring	Interference	1	Prevention	Correctness
Dirty Paper Coding	Interference	1	Prevention	Correctness
Error Correcting Codes	Interference	1	Recovery	Correctness
Error Detection Code (CRC-32)	Interference	2	Detection	Correctness
Additional Checksum (TCP)	Interference	4	Detection	Correctness
CSMA/CD	Collision	2	Prevention, Detection	Correctness, Availability
CSMA/CA	Collision	2	Prevention	Correctness, Availability
Rapid Spanning Tree Protocol	Link Failure	2	Recovery	Availability
Parallel Redundancy Protocol	Link Failure	2.5	Recovery	Availability, Timeliness
Frame Replication and Elimination	Link Failure	2.5	Recovery	Availability, Timeliness
Multipath TCP	Link Failure	4.5	Recovery	Availability
TCP Retransmissions	Data Loss	4	Detection, Recovery	Completeness
WLAN Retransmissions	Data Loss	2	Detection, Recovery	Completeness
TCP Congestion Control	Congestion	4	Prevention, Detection	Availability
TCP Flow Control	Congestion	4	Prevention	Availability
Time Aware Shaping	Congestion	2	Prevention	Availability, Timeliness

TABLE 1: Summary of mechanisms discussed in this paper

as the examples of Ethernet and TCP show. There is not always a particular layer at which failures are addressed.

A mechanism usually targets a specific type of failure. There are several techniques how a failure can be addressed:

1) Prevention. For example, CSMA/CA attempts to prevent collisions.
2) Detection. Error detecting codes, checksums and acknowledgments can be used to detect if a failure occurred.
3) Recovery. These mechanisms attempt to cope with or correct a failure.

The underlying strategy to address a failure depends on the failure itself. Bit errors or link failures are addressed by adding redundancy in one form or another, half-duplex setups attempt to implement time division multiplexing, and congestion is avoided by limiting the amount of traffic in the network.

There is a relationship between the type of failure and the aspect of reliability that is improved. Previously, we defined reliability as complete, correct and timely data transmission, combined with high availability. Interference, link failures and congestion affect the availability of the network and the correctness of the data, so the mechanisms addressing them improve exactly these aspects of reliability. Retransmissions are special in that they target the symptom (= data loss) rather than a specific failure. Regardless of why the data was lost, these mechanisms attempt to recover the data. They are also the only mechanisms capable of improving or ensuring the completeness aspect of reliability.

It can be observed that the timeliness of data transmission requires dedicated mechanisms, since time guarantees are not provided by the common Ethernet/IP/TCP protocol stack. For many applications, a best-effort delivery is good enough, but if some applications have mission-specific Quality of Service (QoS) requirements, additional measures are needed.

4.2. Other Metrics

There are other metrics that can sometimes be used to further describe and compare reliability mechanisms.

Popularity. A very commonly used protocol stack is Ethernet/IP/TCP. This is reasonable, because every type of failure is addressed by at least one protocol. However, since they only provide best-effort data transmission, protocols like FRER and TAS that focus more on QoS have been developed. Also, high popularity does not imply high quality. A memo published by the IEC suggests, that there are CRC-32 polynoms with better properties than the one used in IEEE 802 networks. [27]

Overhead. For most protocols, only a few bytes of additional header fields or some acknowledgment signals are required, but apart from the implementation effort, this does not have too much impact. One influential overhead is hardware cost. While MPTCP can use existing interfaces, PRP requires an investment into dedicated hardware. A paper comparing FRER and PRP suggests that FRER is less expensive to implement. Some mechanisms may only be worth implementing for critical applications, but this is a tradeoff that must be decided for each application. [28]

Flexibility. Hardware setups are not only expensive but also inflexible, e.g. if full physical redundancy is required as with PRP, any host in the network must be connected twice, and at least twice as many network nodes and links have to be maintained. Higher-layer protocols may be more flexible, since they put less requirements on lower layers. They provide general purpose implementations. This may be the reason why TCP has remained incredibly popular since its introduction in 1981.

5. Conclusion

We have defined reliability in the context of communication networks and summarized what types of failures exist. Sixteen different mechanisms that improve the reliability have been described and categorized by the failures that they address and the layer in which they are implemented. Failures affect certain reliability aspects, such as the availability or correctness. Consequently, the mechanisms that address a specific failure improve exactly those aspects of reliability. Reliability mechanisms can prevent, detect, or recover from a failure. Three basic concepts for improving reliability are adding redundancy, providing a multiplexed setup, and not overloading the network capacity.

References

[1] S. Sun, “ACARS Data Identification and Application in Aircraft Maintenance,” in *2009 First International Workshop on Database Technology and Applications*. IEEE, 2009, pp. 255–258.

[2] “Digital Video Broadcasting (DVB); Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2),” p. 22, 2009.

[3] H. Zhang, N. Huang, and H. Liu, “Network performance reliability evaluation based on network reduction,” in *2014 Reliability and Maintainability Symposium*, 2014, pp. 1–6.

[4] R. Li, N. Huang, S. Li, R. Kang, and S. Chang, “Reliability Testing Technology for Computer Network Applications,” in *2009 8th International Conference on Reliability, Maintainability and Safety*, 2009, pp. 1169–1172.

[5] T. Biswas, K. Lesser, R. Dutta, and M. Oishi, “Examining Reliability of Wireless Multihop Network Routing with Linear Systems,” ser. HotSoS ’14. New York, NY, USA: Association for Computing Machinery, 2014. [Online]. Available: https://doi.org/10.1145/2600176.2600195

[6] J. Shi, S. Wang, and K. Wang, “Congestion-Based Reliability Analysis for Computer Metworks,” in *2009 8th International Conference on Reliability, Maintainability and Safety*, 2009, pp. 1149–1154.

[7] “IEEE Standard for Ethernet,” *IEEE Std 802.3-2022 (Revision of IEEE Std 802.3-2018)*, pp. 1–7025, 2022.

[8] “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 802.11-2020 (Revision of IEEE Std 802.11-2016)*, pp. 1–4379, 2021.

[9] “Welcome to the IEEE 802.1 Working Group,” https://1.ieee802.org/, n.d., [Online; accessed 30-November-2022].

[10] “Internet Potocol,” Internet Requests for Comments, RFC Editor, RFC 791, 1981. [Online]. Available: https://www.rfc-editor.org/rfc/rfc791.txt

[11] “Transmission Control Potocol,” Internet Requests for Comments, RFC Editor, RFC 793, 1981. [Online]. Available: https://www.rfc-editor.org/rfc/rfc793.txt

[12] A. Oliviero and B. Woodward, *Cabling: The Complete Guide to Copper and Fiber-Optic Networking*, 4th ed. John Wiley & Sons, 2009.

[13] U. Erez and S. ten Brink, “A Close-to-Capacity Dirty Paper Coding Scheme,” *IEEE Transactions on Information Theory*, vol. 51, no. 10, pp. 3417–3432, 2005.

[14] T. Strutz, “Low-Density Parity-Check Codes - An Introduction,” 2016, [Online; accessed 7-December-2022].

[15] “Internet Control Message Protocol,” Internet Requests for Comments, RFC Editor, RFC 792, 1981. [Online]. Available: https://www.rfc-editor.org/rfc/rfc792.txt

[16] “User Datagram Potocol,” Internet Requests for Comments, RFC Editor, RFC 768, 1981. [Online]. Available: https://www.rfc-editor.org/rfc/rfc768.txt

[17] J. Stone and C. Partridge, “When the CRC and TCP Checksum Disagree,” vol. 30, no. 4. New York, NY, USA: Association for Computing Machinery, aug 2000, p. 309–319. [Online]. Available: https://doi.org/10.1145/347057.347561

[18] L. Georgiadis, *Carrier-Sense Multiple Access (CSMA) Protocols*, 04 2003.

[19] “CSMA/CA: Definition and Explanation of the Method,” 08 2019, [accessed 14-December-2022]. [Online]. Available: https://www.ionos.com/digitalguide/server/know-how/csmaca-carrier-sense-multiple-access-with-collision-avoidance/

[20] W. Wojdak, “Rapid Spanning Tree Protocol: A new solution from an old technology,” pp. 1–5, 2003.

[21] R. Hunt and B. C. Popescu, “Comparison of PRP and HSR Networks for Protection and Control Applications,” 2015.

[22] “IEEE Standard for Local and metropolitan area networks–Frame Replication and Elimination for Reliability,” *IEEE Std 802.1CB-2017*, pp. 1–102, 2017.

[23] D. Ergenç and M. Fischer, “On the Reliability of IEEE 802.1CB FRER,” in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, 2021, pp. 1–10.

[24] G. Noh, H. Park, H. Roh, and W. Lee, “Secure and Lightweight Subflow Establishment of Multipath-TCP,” *IEEE Access*, vol. 7, pp. 1–1, 12 2019.

[25] “802.11 Reference Design: Recovery Procedures and Retransmit Limits,” https://warpproject.org/trac/wiki/802.11/MAC/Lower/Retransmissions, 2014, [Online; accessed 18-December-2022].

[26] M. K. Al-Hares, P. Assimakopoulos, D. Muench, and N. J. Gomes, “Modeling Time Aware Shaping in an Ethernet Fronthaul,” in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, 2017, pp. 1–6.

[27] “Internet Protocol Small Computer System Interface (iSCSI) Cyclic Redundancy Check (CRC)/Checksum Considerations,” Internet Requests for Comments, RFC Editor, RFC 3385, 2002. [Online]. Available: https://www.rfc-editor.org/rfc/rfc3385.txt

[28] G. Ditzel, “The Comparison/Contrast of TSN Frame Replication and Elimination for Reliability (FRER) and Parallel Redundancy Protocol (PRP),” pp. 1–13, 2020.

Survey on the Chinese Governments Censorship Mechanisms

Raphael Stadler, Lion Steger*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: r.stadler@tum.de, stegerl@net.in.tum.de

***Abstract*—The Chinese government enforces a strict censorship policy on digital content and has created an isolated network for its residents. As it is not feasible to completely disconnect China from the global internet, the government has implemented a powerful and complex system called the Great Firewall of China (GFW) to separate the national network from the rest of the world. This paper focuses on how the Chinese government implements its censorship policies and outlines various techniques the GFW uses to block specific traffic. Subsequently, we present numerous ways on how it is possible to bypass these methods to illustrate the ongoing conflict between the GFW and anti-censorship communities.**

***Index Terms*—great firewall of china, gfw, gfc, golden shield project, dns poison, sni filtering, shadowsocks, v2ray, tor, fronting**

1. Introduction

The Great Firewall of China (GFW), also known as The Golden Shield project [1], is a government controlled firewall that acts not only inside the countries network, but also at the interconnections between the national Chinese network and the global networks, the internet. It resides in-between every connection that is initiated to or from China, similar to an attacker like "Man or Machine in the Middle" (MitM). With this powerful position in the network, the GFW can not only observe and evaluate every connection passively, but it can also act actively by modifying connections or operating maliciously in-between two peers, which is necessary for fulfilling the governments policies. Figure 1 demonstrates the position of the GFW and shows that practically no connection can be initiated without the GFW in the middle.

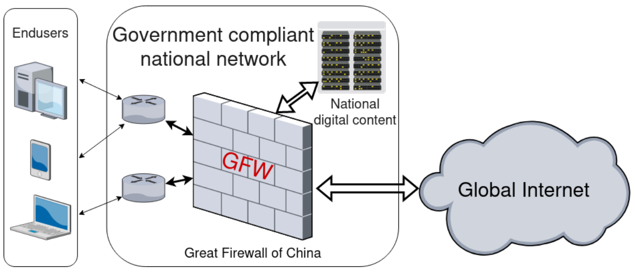


Figure 1: A schematic illustration of the Great Firewall of China and its reach in the Chinese network.

2. Overview of various blocking methods

The GFW has used and is using various means to censor, filter or prevent unwanted traffic. As there are several benefits as well as disadvantages on each method presented below, the practical implementation of the GFW adjusts its sensors dynamically in complex and obscure manners. The requirement for this is driven by the ongoing evolution of technology and the potential for changes in bypassing methods.

2.1. Subnet Blocking and Re-routing

One of the simplest methods the GFW is using is blocking whole IP subnets [2]. This can be done by modifying routing tables and re-routing specific IP subnets. These altered routes either redirect traffic to GFW controlled servers trying to mimic the actual target service, and therefore may gain data for further analysis, or they can be null routed, which means that effectively any connection made to the destination is prevented. Alternatively, the GFW can accomplish similar results by using hijacking Border Gateway Protocol (BGP) sessions. BGP hijacking works by maliciously announcing unowned and improper BGP prefixes. While these altered BGP prefixes are normally announced in the national networks only, it does happen that announcements also get accepted by international networks as well, if they are announced incorrectly or with harmful intents. [3]

Both methods can lead to collateral damage, as IP addresses might be shared by various services, which are then blocked as well. IP blocks or BGP hijacking are commonly used by the GFW, as they allow to block a service quickly and effectively. These blocks are often based on research, such as connection analysis or publicly available information. The GFW blocks have cause great collateral damage [4], [5], which is another reason why the GFW implements a fairly dynamical concept and lifted various blocks again.

2.2. DNS Poisoning

Another method of blocking or redirecting traffic is *DNS Poisoning*. When using a DNS resolver inside of the Chinese network, it must obey the law and enforce the governments policies. A DNS request that contains a blocked hostname resolves to a different IP address than internationally announced. [6] The GFW uses a more sophisticated approach to censor internet access for resolvers that are outside of the Chinese network. It lets the request resolve correctly and leaves it untouched in the outgoing direction and sends a crafted response back to the client that initiated the connection. The GFW spoofs the DNS resolver's IP and exchanges the payload. This is possible as there is no authentication or encryption in the DNS protocol. [7] While possible to filter the actual response, the GFW allows, to some extent, the original response to pass through, which leads to race conditions on which DNS reply is faster. One example of DNS poisoning happened in 2013, when the website *github.com* was blocked in China. [4] Later it was lifted again due to an unforeseen amount of protest in the Chinese community. Another example where DNS packets were inspected occurred in 2002 when *web.mit.edu* was filtered due to resolving to the same IP address as another hostname *www.falundafa.org*, which had been banned by the Chinese government. [5]

2.3. Encrypted DNS

DNS Poisoning can be bypassed by using encrypted DNS requests with implementations such as DNS-over-HTTPS (DoH), DNS-over-TLS (DoT), DNS-over-QUIC (DoQ) or DNSCrypt. These protocols require that the resolver remains outside of the censored network to work as intended. [7] The encrypted nature of some network requests makes it difficult for the GFW to analyse or modify the packets. The GFW has shown that it can block encrypted DNS requests, which makes these protocols only partially effective as bypassing strategies [8]. Encrypted DNS requests are not a reliable solution to bypass DNS poisoning, if the resolver is controlled by the GFW. In fact, using a DoH resolver operated by Alibaba can yield similar incorrect responses as using unencrypted Chinese resolvers. Even though the encryption has not been broken, it suggests that the resolvers operated by Alibaba are at least working in cooperation with the GFW. [8]

2.4. Keyword Filtering

The GFW is able to detect keywords in network requests by inspecting traffic and terminates connections accordingly. [9] This process is similar to DNS poisoning, where an unwanted term can be identified in a plain and unencrypted request. While not limited to the TCP protocol, a good indicator for this mechanism can be observed in TCP-based traffic, as the GFW reacts by sending a *TCP-RST* packet back on the same channel, terminating any further communication between two peers. [10] This method of identification and reaction from the GFW is practically possible on every protocol transmitting plaintext, such as SMTP, IMAP, POP or TELNET. Due to the prevalence of HTTP-based traffic, the GFW has developed specialized capabilities for detecting keywords in HTTP-based traffic. [11] When a user attempts to access a website, the GFW's keyword filtering system inspects the URL or the content of the page to see if it contains any keywords that are on a pre-defined list of prohibited words or phrases. If the website or page contains prohibited keywords, it will block the request and prevent the user from accessing the website. The GFW is especially enforcing this technique in Chinese search engines, such as *google.cn* or *baidu.cn* [12]

2.5. SNI-Filtering

With an increase of implementing TLS on web-servers [2] and using encrypted connections, this approach is ineffective if the corresponding target server does not comply to the Chinese governments policies by letting the GFW inspect the unencrypted content. While companies inside of China often cooperate with the government [13], this is mostly the case for services operating outside of China. As IP addresses are often shared, a technique called *Server Name Indication* (SNI) is currently widely implemented to serve encrypted content for multiple hostnames. SNI helps in identifying which specific service is to be provided by sending the requested hostname in plaintext, as it is required and sent during the unencrypted connection setup. This allows the GFW to filter connections based on the hostname, while minimizing collateral damage and allowing other hostnames to continue using the IP address. The detection rate on such connections is limited to hostname filtering. This vulnerability of SNI can be remedied by using encrypted SNI. ESNI works by first retrieving a trusted encryption key and sending the encrypted SNI header to the web-server. It was further investigated, that the GFW filters ESNI connections currently, purely by dropping detected packets. [2] It blocks only a certain ESNI version, while leaving other protocol implementations with other versions untouched. [14]

2.6. Other detection methods

Similarly to ESNI, the GFW has difficulties detecting and verifying content when the *QUIC* protocol is used, due to the implicit encryption. Currently, blocks are less stringent and QUIC requests are mostly passing through. It is expected that QUIC packets might be dropped more aggressively or that the GFW maintainers implement inspection algorithms, if possible. [15]

Another method the GFW implements is bandwidth throttling. While this is not a practical way of blocking connections, it discomforts the use of bypassing methods for end-users, which might achieve the effect that they give up or reduce their use of bypassing methods. Especially for international connections, throttling is generally in place. [16]

While not commonly linked to casual internet connections, using torrents allows accessing and downloading files as well. It was shown, that one of the biggest Chinese ISPs, namely China Telecom, does prevent torrenting, likely in compliance to the Chinese governments strategy. [17]

2.7. Combining methods

The GFW is capable to implement multiple methods to better enforce their blocking strategy. For example, while bypassing DNS poisoning and receiving the correct IP address when using some DNS bypassing technique, the GFW can still block connections to the target, if HTTP would be used. A scenario was shown, where even using connections based on HTTPS with enabled SNI would be blocked solely by the leaked hostname, despite bypassing the DNS poisoning of the GFW. [8]

3. General bypassing methods

One great drawback of previously discussed bypassing methods is the demand of specific implementations on the content delivering server. As an end-user normally has no control over such single services, it is beneficial for them to rely on some form of general circumvention technique for explicitly selected or all connections. The systems presented below work by sending encapsulated traffic to a middle server, which then unwraps the original request and executes it in position for the end-user.

3.1. Bypassing via proxies

In general a technical server proxy is a piece of software, that acts as a gateway between two machines. Proxies can act on different layers in the OSI/ISO motel, the current common services however mostly act on the highest layers, for instance using HTTP and HTTPS (Layer 7) or the SOCKS protocol (Layer 5). Some common software that is used for proxies are shadowsocks, obfs (by TOR), Trojan or V2Ray. [18]–[20] A proxy, by design, needs to be configured for the specific software that should use it. Meaning, that it normally is only used by an explicit application which then encapsulates it's requests into the proxy specific protocol.

3.2. Bypassing via VPNs

Another common method is to use virtual private networks (VPNs). Despite that the general idea of a VPN is not meant for utilizing it to bypass restrictive censorship, it is a practical utility for doing so. [21] VPNs, while very similar to proxies, normally act on lower layers, by using implementations such as OpenVPN, WireGuard or IPSec. They therefore have more overhead in comparison to proxies but can route traffic of all applications through the tunnel. This is specifically useful for application that do not support proxies. [22]

3.3. Other bypassing methods

Another viable option for end-users to use is a *Web Proxy*. [23], [24] While quite similar to a regular proxy in terms of functionality, a web proxy is an application hosted on a web-server, that is only designed to allow users to browse websites via this specific server. [25]

Using international SIM cards is another possible solution to bypass censorship, as this specific traffic is routed to the mobile network service provider and therefore normally not filtered at all. To reduce latency, SIM cards from ISPs in Hong Kong are currently the most favoured by the Chinese communities.

4. Detection and verification of circumvention services

Significant efforts have been made to bypass the GFW, but many systems encounter challenges in the identification of connections based solely on recognizable properties. Passive detection of connections relies on specific identifiers, such as port and protocol. When using standard software configurations, fixed ports are often used, enabling the easy identification of connections. By default, OpenVPN is listening on UDP port 1194 and IPSec uses UDP and TCP on specific ports and the rather ESP protocol. As a result, it is straightforward to detect and block such connections for the GFW. [26], [27] Additionally, it is possible to further detect specific VPN software implementations based on a technique called *Fingerprinting*.

4.1. OpenVPN

Especially as OpenVPN is currently widely used in commercial VPN software, it is a high priority target for the GFW maintainers. [28] While some identifiers are based on publicly available data, such as WHOIS information, AS numbers of VPN providers or PTR records of their IP addresses, it was shown that it is possible to detect OpenVPN traffic based on properties such as the unencrypted operational byte patterns in encapsulated packets. Although this detection technique already has a high accuracy of above 99% while only needing ten connection packets, it is also possible to recognize TCP based OpenVPN traffic on other properties, such as specific TCP options. [29] Although there are some proposed protocol changes that would obfuscate OpenVPN-based traffic, some are disregarded by the OpenVPN developers, such as an XOR scrambler. Others often rely on different obfuscation protocols such as shadowsocks, obfs or even proprietary protocols, all with their own disadvantages. [28] OpenVPN is currently not useful in bypassing very strictly censored networks, such as the national networks in China.

4.2. TOR and obfs

Another detection method uses fingerprinting techniques for TOR-based traffic. Initially, the GFW blocked TOR by using publicly available information on TOR entry servers. The TOR community responded by introducing unpublished TOR bridges, but the GFW was still able to detect these bridges by identifying the used ciphers. Trying to mitigate this by using obfuscators, it was still possible for the GFW to detect bridges by using *Active Probing*. Active probing involves scanning for keywords on third-party service providers such as websites, forums, or similar platforms, and analysing VPN or proxy services for the purpose of blocking them. In the case of TOR-based traffic, if the GFW detected a connection with a likely chance of being used in TOR related traffic, it then would send specifically crafted packets to the suspected TOR bridge and confirm and block the IP address, if it was responding as expected. It was shown that the GFW could block an unpublished TOR bridge practically instantly after the first connection was initiated from an end-user in China. [30]

4.3. Shadowsocks

Shadowsocks-based proxies with early implementations were found to be susceptible to detection due to a lack of authentication. [19] By sending specific packet headers to a proposed server, an attacker could exploit, that a proxy would respond. [22] With this response, the GFW could then verify that it was a proxy service and therefore block connections to the IP address. This attack could partially be mitigated by implementing an OTA (One-Time Auth) mechanism on the payload. A later improved protocol specification using AEAD ciphers addressed this vulnerability in shadowsocks. [31]

5. Mimicking and Fronting

To bypass the GFW, VPN or proxy service operators often try to mimic commonly used services and therefore reduce the risk of detection. HTTPS is an excellent protocol for such concealment, as very much of the traffic on the internet is HTTP/HTTPS based. [11] Additionally, HTTPS traffic is already encrypted, which allows to easily reuse the protocols properties for building encrypted tunnels. This is also one reason, why many tunnels specifically designed for bypassing strict censorship do mimic HTTPS connections, currently with a notable success rate.

To further prevent blocks, bypassers often use *Fronting* to increase collateral damage if blocks are actually enforced. When implementing it, another provider is put in front of the actually bypassing service. For instance, when setting up a private VPN service, it helps to host such service on popular cloud service providers, such as Amazon Web Services. With those in place, the GFW could not ban the whole network, as legitimate services are operating on the same platforms as well. Single IPs can however still be blocked.

5.1. Importance of Content Delivery Networks

Therefore, it is a better method to use Content Delivery Networks (CDNs) as a fronting mechanism. As much of the HTTPS traffic is cached and accelerated using commercial CDNs, they can also be used in fronting for tunnel services as well, if the tunnel uses HTTPS. For instance, at the time of this research, one of the most recent tunnelling setups would not only include the combination of a CDN and a HTTPS (forward) proxy, namely V2Ray, but also a HTTPS reverse proxy (nginx) to allow custom webserver content to be hosted on the same machine as well. [32] [33] This reverse proxy helps in mitigating blocks, as there might be actual content displayed if visiting the website normally and therefore could look like a normal webpage. To the GFW, it is therefore much harder to block the connection, as it appears to be regular HTTPS traffic, and because the IP address of the actual proxy is hidden behind the CDN's IP address. [34] [35]

These benefits apply to web proxies as well, as they have the great advantage of being used in combination with highly advanced fronting methods easily. Despite the limitations, it might remain one of the most undetected proxying methods, as the application does not look different to normal content hosted on web-servers and further, as there are a lot different implementations from different creators as well.

The GFW's current attempts to prevent these setups is done by throttling international CDNs and forcing companies to setup their infrastructure in China as well. [16] These servers in China must obey the law, which allows the government to access data or enforce their own policies on these companies relatively easily. Companies, such as *Apple Inc.*, censor content even without the Chinese government interfering officially. [13]

6. Verification of GFW's blocking techniques

In the course of this project, some experiments to test the GFW's capabilities from outside of the Chinese censored network have been setup. This works, because the GFW often acts in a symmetric way.

We could prove the existence of DNS poisoning by sending DNS queries to Chinese based IP addresses. Trying to resolve AAAA records returned invalid IPv6 addresses, A record requests were responded with random but routable IPv4 addresses. It should be noted that the GFW intercepts packets before they reach their intended destination, eliminating the need for the targeted IP addresses to provide DNS resolution services, which is an additional indicator for DNS poisoning if requests are not intercepted by the GFW. To compare, we further queried multiple Chinese based resolvers and other international resolvers and analysed the response data.

TABLE 1: HTTP-filtering results

Hostname	Target IP	Filtering	Reason
alibaba.cn	140.205.174.2		
baidu.cn	220.181.38.148		
facebook.com	**140.205.174.2**	x	TCP-RST
wikipedia.org	**140.205.174.2**	x	TCP-RST
baidu.cn	**140.205.174.2**		HTTP Errors

Another blocking technique we tried to verify was keyword filtering. For this experiment, demonstrated in Table 1, we made use of the command-line tool *curl* to actively resolve a hostname to a Chinese IP for crafted HTTP requests. When connecting to the valid Chinese IP, such as the IP *140.205.174.2* that belongs to *alibaba.cn*, connection resets (TCP-RST) packets could always be observed when we tried to connect to it by using various known blocked hosts, such as *facebook.com* or *wikipedia.org*. When using allowed hosts in the hostname, such as *baidu.cn* or *google.cn* the connection was established and HTTP content was transmitted correctly, leading to expected HTTP error codes.

TABLE 2: SNI-Filtering results

Hostname	Target IP	Filtering	Reason
alibaba.cn	140.205.174.2		
baidu.cn	220.181.38.148		
wikipedia.org	**140.205.174.2**	x	TCP-RST
facebook.com	**140.205.174.2**	x	TCP-RST
baidu.cn	**140.205.174.2**		TLS Mismatch

Furthermore, similar results could be achieved for HTTPS requests as illustrated in Table 2. For *facebook.com*, *wikipedia.org* and others, the TLS handshake was interrupted due to TCP-RSTs. When using invalid hostnames, such as *baidu.cn* or *tencent.cn*, a TLS certificate mismatch could be conducted. This expected result suggests that the connection was not interrupted and the TLS handshake completed as expected. When connecting with the actual hostname *alibaba.cn* content was transmitted as expected.

7. Conclusion and future work

The Great Firewall of China is a very powerful tool in enforcing the strict censorship policies the Chinese government sets up. It is a strong counterpart to the freedom of speech communities by demonstrating the possibilities and techniques it can implement and execute in the Chinese national network. However, the GFW is not perfect and there is much work done to bypass it, not only from the inside of China but also internationally. Especially when combining multiple bypassing methods and with the development of new designs and protocols, the Chinese government needs to perfectly balance between isolating its citizens from global information and separate itself too much from the rest of the world.

References

[1] S. Chandel, Z. Jingji, Y. Yunnan, S. Jingyao, and Z. Zhipeng, “The Golden Shield Project of China: A Decade Later—An in-Depth Study of the Great Firewall,” in *2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, Oct. 2019, pp. 111–119.

[2] Z. Chai, A. Ghafari, and A. Houmansadr, “On the Importance of Encrypted-SNI (ESNI) to Censorship Circumvention,” p. 8.

[3] C. Demchak and Y. Shavitt, “China’s Maxim – Leave No Access Point Unexploited: The Hidden Story of China Telecom’s BGP Hijacking,” *Military Cyber Affairs*, vol. 3, no. 1, Jun. 2018, accessed on 2022-11-21. [Online]. Available: http://scholarcommons.usf.edu/mca/vol3/iss1/7/

[4] S. N. Company, “Sudo Null - Latest IT News,” accessed on 2022-11-21. [Online]. Available: https://sudonull.com/post/132004-Github-is-blocked-in-China

[5] “China Blocks MIT Web Addresses - The Tech,” accessed on 2022-11-21. [Online]. Available: http://tech.mit.edu/V122/N58/58web.58n.html

[6] O. Farnan, A. Darer, and J. Wright, “Poisoning the Well: Exploring the Great Firewall’s Poisoned DNS Responses,” in *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society*. Vienna Austria: ACM, Oct. 2016, pp. 95–98, accessed on 2022-11-21. [Online]. Available: https://dl.acm.org/doi/10.1145/2994620.2994636

[7] C. Kwan, P. Janiszewski, S. Qiu, C. Wang, and C. Bocovich, “Exploring Simple Detection Techniques for DNS-over-HTTPS Tunnels,” in *Proceedings of the ACM SIGCOMM 2021 Workshop on Free and Open Communications on the Internet*. Virtual Event USA: ACM, Aug. 2021, pp. 37–42, accessed on 2022-11-21. [Online]. Available: https://dl.acm.org/doi/10.1145/3473604.3474563

[8] L. Jin, S. Hao, H. Wang, and C. Cotton, “Understanding the Impact of Encrypted DNS on Internet Censorship,” in *Proceedings of the Web Conference 2021*. Ljubljana Slovenia: ACM, Apr. 2021, pp. 484–495, accessed on 2022-11-21. [Online]. Available: https://dl.acm.org/doi/10.1145/3442381.3450084

[9] Z. Weinberg, D. Barradas, and N. Christin, “Chinese Wall or Swiss Cheese? Keyword filtering in the Great Firewall of China,” in *Proceedings of the Web Conference 2021*, ser. WWW ’21. New York, NY, USA: Association for Computing Machinery, Jun. 2021, pp. 472–483, accessed on 2022-11-30. [Online]. Available: https://doi.org/10.1145/3442381.3450076

[10] J. Zittrain and B. Edelman, “Internet filtering in China,” *IEEE Internet Computing*, vol. 7, no. 2, pp. 70–77, Mar. 2003, conference Name: IEEE Internet Computing.

[11] R. Fontugne, P. Abry, K. Fukuda, D. Veitch, K. Cho, P. Borgnat, and H. Wendt, “Scaling in Internet Traffic: A 14 Year and 3 Day Longitudinal Study, With Multiscale Analyses and Random Projections,” *IEEE/ACM Transactions on Networking*, vol. 25, no. 4, pp. 2152–2165, Aug. 2017, accessed on 2022-11-24. [Online]. Available: http://ieeexplore.ieee.org/document/7878657/

[12] J. S. O’Rourke, B. Harris, and A. Ogilvy, “Google in China: government censorship and corporate reputation,” *Journal of Business Strategy*, vol. 28, no. 3, pp. 12–22, May 2007, accessed on 2022-11-21. [Online]. Available: https://www.emerald.com/insight/content/doi/10.1108/02756660710746229/full/html

[13] S. Liao, “Apple officially moves its Chinese iCloud operations and encryption keys to China,” Feb. 2018, accessed on 2022-12-01. [Online]. Available: https://www.theverge.com/2018/2/28/17055088/apple-chinese-icloud-accounts-government-privacy-speed

[14] “Exposing and Circumventing China’s Censorship of ESNI,” Aug. 2020, accessed on 2022-11-21. [Online]. Available: https://geneva.cs.umd.edu/posts/china-censors-esni/esni/

[15] K. Elmenhorst, B. Schütz, N. Aschenbruck, and S. Basso, “Web censorship measurements of HTTP/3 over QUIC,” in *Proceedings of the 21st ACM Internet Measurement Conference*. Virtual Event: ACM, Nov. 2021, pp. 276–282, accessed on 2022-12-01. [Online]. Available: https://dl.acm.org/doi/10.1145/3487552.3487836

[16] P. Zhu, K. Man, Z. Wang, Z. Qian, R. Ensafi, J. A. Halderman, and H. Duan, “Characterizing Transnational Internet Performance and the Great Bottleneck of China,” *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 4, no. 1, pp. 1–23, May 2020, accessed on 2022-12-01. [Online]. Available: https://dl.acm.org/doi/10.1145/3379479

[17] M. Dischinger, A. Mislove, A. Haeberlen, and K. P. Gummadi, “Detecting bittorrent blocking,” in *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement conference - IMC ’08*. Vouliagmeni, Greece: ACM Press, 2008, p. 3, accessed on 2022-11-21. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1452520.1452523

[18] P. Liubinskii, “The Great Firewall’s active probing circumvention technique with port knocking and SDN,” p. 65.

[19] X. Zeng, X. Chen, G. Shao, T. He, Z. Han, Y. Wen, and Q. Wang, “Flow Context and Host Behavior Based Shadowsocks’s Traffic Identification,” *IEEE Access*, vol. 7, pp. 41 017–41 032, 2019, conference Name: IEEE Access.

[20] Z. Deng, Z. Liu, Z. Chen, and Y. Guo, “The Random Forest Based Detection of Shadowsock’s Traffic,” in *2017 9th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, vol. 2, Aug. 2017, pp. 75–78.

[21] P. Ferguson and G. Huston, “What is a VPN - Univ-pau.fr,” Apr. 1998, accessed on 2022-12-01. [Online]. Available: https://cpham.perso.univ-pau.fr/ENSEIGNEMENT/COMMUN/vpn_ferguson.pdf

[22] Alice, Bob, Carol, J. Beznazwy, and A. Houmansadr, “How China Detects and Blocks Shadowsocks,” in *Proceedings of the ACM Internet Measurement Conference*. Virtual Event USA: ACM, Oct. 2020, pp. 111–124, accessed on 2022-11-24. [Online]. Available: https://dl.acm.org/doi/10.1145/3419394.3423644

[23] “PHP Online Web Proxy,” Nov. 2022, accessed on 2022-11-24. [Online]. Available: https://github.com/NicheOffice/php-web-proxy

[24] Athlon1600, “php-proxy-app,” Nov. 2022, accessed on 2022-11-24. [Online]. Available: https://github.com/Athlon1600/php-proxy-app

[25] J. Dick, “A simple PHP web proxy.” Nov. 2022, accessed on 2022-11-24. [Online]. Available: https://github.com/joshdick/miniProxy

[26] A. Cain and A. Proctor, “OpenVPN Access Server System Administrator Guide,” p. 58.

[27] K. Seo and S. Kent, “Security Architecture for the Internet Protocol,” Internet Engineering Task Force, Request for Comments RFC 4301, Dec. 2005, accessed on 2022-12-01. [Online]. Available: https://datatracker.ietf.org/doc/rfc4301

[28] D. Xue, R. Ramesh, A. Jain, M. Kallitsis, J. A. Halderman, J. R. Crandall, and R. Ensafi, “OpenVPN is Open to VPN Fingerprinting,” p. 19.

[29] Y. Pang, S. Jin, S. Li, J. Li, and H. Ren, “OpenVPN Traffic Identification Using Traffic Fingerprints and Statistical Characteristics,” in *Trustworthy Computing and Services*, Y. Yuan, X. Wu, and Y. Lu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 443–449.

[30] A. Dunna, C. O'Brien, and P. Gill, "Analyzing China's Blocking of Unpublished Tor Bridges."

[31] "Shadowsocks active-probing attacks and defenses," accessed on 2022-11-21. [Online]. Available: https://groups.google.com/g/traffic-obf/c/CWO0peBJLGc/m/Py-clLSTBwAJ

[32] "V2Ray (WebSocket + TLS + Web + Cloudflare) - Eric," accessed on 2022-11-21. [Online]. Available: https://ericclose.github.io/V2Ray-TLS-WebSocket-Nginx-with-Cloudflare.html

[33] "Use Cloudflare Certificate with Nginx + V2Ray + Websocket +TLS + CDN," Aug. 2020, accessed on 2022-11-21. [Online]. Available: https://henrywithu.com/use-cloudflare-certificate-with-nginx-v2ray-websocket-tls-cdn/

[34] S. Satija and R. Chatterjee, "BlindTLS: Circumventing TLS-based HTTPS censorship," in *Proceedings of the ACM SIGCOMM 2021 Workshop on Free and Open Communications on the Internet*. Virtual Event USA: ACM, Aug. 2021, pp. 43–49, accessed on 2022-11-21. [Online]. Available: https://dl.acm.org/doi/10.1145/3473604.3474564

[35] D. Fifield, C. Lan, R. Hynes, P. Wegmann, and V. Paxson, "Blocking-resistant communication through domain fronting," *Proceedings on Privacy Enhancing Technologies*, vol. 2015, no. 2, pp. 46–64, Jun. 2015, accessed on 2022-11-21. [Online]. Available: https://petsymposium.org/popets/2015/popets-2015-0009.php

ISBN 978-3-937201-77-1
DOI 10.2313/NET-2023-06-1

ISSN 1868-2634 (print)
ISSN 1868-2642 (electronic)