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

Proceedings of the Seminars Future Internet (FI) and Innovative Internet Technologies and Mobile Communication (IITM)

Winter Semester 2017/2018

Munich, Germany

Editors

Georg Carle, Daniel Raumer

Publisher

Chair of Network Architectures and Services

Lehrstuhl für Netzarchitekturen und Netzdienste
Fakultät für Informatik
Technische Universität München

Proceedings zu den Seminaren Future Internet (FI) und Innovative Internet-Technologien und Mobilkommunikation (IITM)

Wintersemester 2017/2018

München, 1. 8. 2017 – 26. 2. 2018

Editoren: Georg Carle, Daniel Raumer

Network Architectures
and Services
NET 2018-03-1

Proceedings of the Seminars
Future Internet (FI) and Innovative Internet Technologies and Mobile Communication (IITM)
Winter Semester 2017/2018

Editors:

Georg Carle
Lehrstuhl für Netzarchitekturen und Netzdienste (I8)
Technische Universität München
85748 Garching b. München, Germany
E-mail: carle@net.in.tum.de
Internet: `https://net.in.tum.de/~carle/`

Daniel Raumer
Lehrstuhl für Netzarchitekturen und Netzdienste (I8)
E-mail: raumer@net.in.tum.de
Internet: `https://net.in.tum.de/~raumer/`

Cataloging-in-Publication Data

Seminars FI & IITM WS 17/18
Proceedings zu den Seminaren „Future Internet“ (FI) und „Innovative Internet-Technologien und Mobilkommunikation“ (IITM)
München, Germany, 1. 8. 2017 – 26. 2. 2018
ISBN: 978-3-937201-61-0

ISSN: 1868-2634 (print)
ISSN: 1868-2642 (electronic)
DOI: 10.2313/NET-2018-03-1
Lehrstuhl für Netzarchitekturen und Netzdienste (I8) NET 2018-03-1
Series Editor: Georg Carle, Technische Universität München, Germany

Vorwort

Vor Ihnen liegen die Proceedings der beiden Seminare „Future Internet“ (FI) und „Innovative Internet-Technologien und Mobilkommunikation“ (IITM), welche die finalen Ausarbeitungen unserer Studierenden enthalten. Die beiden Seminare wurden am Lehrstuhl für Netzarchitekturen und Netzdienste an der Fakultät für Informatik der Technischen Universität München im Wintersemester 2017/2018 durchgeführt. Allen Teilnehmerinnen und Teilnehmern stand es wie in der Vergangenheit frei, die Ausarbeitung und den Vortrag in englischer oder in deutscher Sprache zu verfassen. Dementsprechend finden sich sowohl englische als auch deutsche Beiträge in diesen Proceedings.

Unter den Teilnehmern der Seminare verliehen wir, wie in jedem Semester, jeweils einen Best Paper Award. Die beste Arbeit im FI-Seminar wurde von Herrn Jakub Wójcik verfasst, der in seiner Ausarbeitung „Practical Assessment of Secure Multiparty Computation Frameworks“, eine Übersicht über existierende Frameworks zur Secure Multiparty Computation gibt. Die Arbeit liefert einen Vergleich der Frameworks und füllt eine Lücke fehlender Dokumentation in diesem jungen Forschungsfeld. Im IITM-Seminar ging dieser an Herr Jan Luca Pawlik, der in seiner Ausarbeitung „Kryptocoin Diebstähle“ einen erschöpfenden Überblick über Angriffe auf Kryptocoinnetzwerke und eine Taxonomie dazu etabliert.

Einige der Vorträge wurden aufgezeichnet und sind auf unserem Medienportal unter `https://media.net.in.tum.de` abrufbar.

Im FI-Seminar wurden Beiträge zu den folgenden Themen verfasst:

- Videostreamerkennung und QoE-Abschätzung in verschlüsselten Verbindungen
- Overhead von IoT-Kommunikationsprotokollen
- Analyse von Secure Multiparty Computation Frameworks

Auf `https://media.net.in.tum.de/#%23Future%20Internet%23WS17` können die aufgezeichneten Vorträge zu diesem Seminar abgerufen werden.

Im IITM-Seminar wurden die folgenden Themen abgedeckt:

- Ethik, Produkte, Top-Listen – und ihre Benutzung Konferenzen für Internetmessungen
- Text Mining auf Mailinglisten: Tagging
- Kryptocoin Diebstähle
- WeSee: Dynamische Visualisierung von Webserviceauslastungen
- Hybridansätze von neuronalen Netzen und genetischen Algorithmen

Auf `https://media.net.in.tum.de/#%23IITM%23WS17` können die aufgezeichneten Vorträge zu diesem Seminar abgerufen werden.

Wir hoffen, dass Sie den Beiträgen dieser Seminare wertvolle Anregungen entnehmen können. Falls Sie weiteres Interesse an unseren Arbeiten haben, so finden Sie weitere Informationen auf unserer Homepage `https://net.in.tum.de`.

München, März 2018

Georg Carle

Daniel Raumer

Preface

We are pleased to present you the proceedings of our seminars on "Future Internet" (FI) and "Innovative Internet Technologies and Mobile Communication" (IITM) which took place in the Winter Semester 2017/2018. In all seminar courses organized by our research group, the authors were free to write their paper and give their talk in English or German.

We honoured the best paper of each seminar with an award. This semester the award in the FI seminar was given to Mr. Jakub Wójcik for his paper "Practical Assessment of Secure Multiparty Computation Frameworks" wherein he presents an overview to existing secure multiparty computation frameworks. In the IITM seminar the award was given to Mr. Jan Luca Pawlik who analysed crypto coin thefts in his paper "Kryptocoin Diebstähle".

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

In the seminar FI we dealt with issues and innovations in network research. The following topics were covered:

- Streaming Video Detection and QoE Estimation in Encrypted Traffic
- Comparison of IoT Data Protocol Overhead
- Practical Assessment of Secure Multiparty Computation Frameworks

Recordings can be accessed on `https://media.net.in.tum.de/#%23Future%20Internet%23WS17`.

In the seminar IITM we dealt with different topics in the area of network technologies, including mobile communication networks. The following topics were covered:

- Ethics, Products, Top Lists – and their Use at Internet Measurement Conferences
- Text Mining on Mailing Lists: Tagging
- Kryptocoin Thefts
- WeSee: dynamic visualization of Web Service use
- Hybridization of Neural Networks and Genetic Algorithms

Recordings can be accessed on `https://media.net.in.tum.de/#%23IITM%23WS17`.

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, March 2018

Seminarveranstalter

Lehrstuhlinhaber

Georg Carle, Technische Universität München, Germany

Seminarleitung

Daniel Raumer, Technische Universität München, Germany

Betreuer

Marcel von Maltitz (maltitz@net.in.tum.de)
Technische Universität München

Kajó Márton (kajo@in.tum.de)
Technische Universität München

Heiko Niedermayer (niedermayer@net.in.tum.de)
Technische Universität München

Miguel L. Pardal (miguel.pardal@tecnico.ulisboa.pt)
Instituto Superior Técnico, Universidade de Lisboa

Daniel Raumer (raumer@net.in.tum.de)
Technische Universität München

Quirin Scheitle (scheitle@net.in.tum.de)
Technische Universität München

Jan Seeger (seeger@net.in.tum.de)
Technische Universität München

Seminarhomepage

`https://net.in.tum.de/teaching/ws1718/seminars/`

Inhaltsverzeichnis

Future Internet

Innovative Internet-Technologien und Mobilkommunikation

Streaming Video Detection and QoE Estimation in Encrypted Traffic

Bogdan Iacob
Advisor: Kajó Márton
Seminar Future Internet WS2017/2018
Chair of Network Architectures and Services
Departments of Informatics, Technical University of Munich
Email: bogdan.iacob@in.tum.de

ABSTRACT

With security and privacy as key matters of evolving importance in the field of online content distribution, Over The Top (OTT) providers of video streaming services shifted their focus towards end-to-end encryption. At the same time, the demand for online video content has been dramatically increasing, as a result of the ever-growing number of users and the expanding popularity of video streaming services. While end users benefit from the improvement in privacy, the change to encrypted video traffic and the high market demands raise new challenges for Internet Service Providers in the quest of monitoring service performance and maintaining a competitive level of quality.
This paper describes current research that addresses the challenges of detecting streamed video and estimating the Quality of Experience (QoE) in encrypted traffic. In approaching these challenges, various methodologies, relying on network traffic analysis and machine learning-based QoE classification models, are presented.

Keywords

Encrypted traffic; Network measurements; Video streaming; Quality of Experience; Passive monitoring; Machine learning

1. INTRODUCTION

The need for improved online security increased substantially side by side with the growth of the Internet. As a result, the expansion of HTTPS across the World Wide Web has gained a strong momentum. Along this trend, big players in the online video streaming industry switched to encrypted traffic in delivering their video services. One of the biggest OTT providers of video streaming services, YouTube, began the process in 2011 and managed to roll out encryption using HTTPS to 97 percent of its total traffic by 2016 [18]. In a similar manner, Netflix's Internet television network switched to encrypted traffic in 2016 [16].
Furthermore, Cisco forecasts the global IP video traffic to increase at a rapid pace in the next 4 years, to an expected 82 percent of all consumer Internet traffic by 2021 [2].
Consequently, the importance of good QoE estimations has risen considerably, while the QoE management has become an even bigger challenge for ISPs and mobile operators, which rely their measurements mostly on deep packet inspection [16].
QoE is needed in order to allow ISPs and mobile operators to assess and improve the quality of their services in order to remain competitive.

The end-user QoE is determined based on the evaluation of certain Key Performance Indicators (KPIs), such as initial start-up delay, stalling, average video quality in terms of bitrate and resolution, as well as quality variations. While these KPIs could be easily extracted from video traffic based on HTTP, by performing passive network measurements, the use of HTTPS prevents the inspection of packet headers and payload data. In attempt to address this challenge, research presented in this paper proposes approaches based on Machine Learning algorithms to statistically relate network- to application-level metrics, as a solution to detect QoE degradation.
The rest of the paper folds into 3 main chapters. Chapter 2 presents current mechanisms for detecting streamed videos and provides a brief analysis with regard to their performance and accuracy, as well as their impact on the Net Neutrality principles. Chapter 3 approaches the process of QoE estimation, describing the KPIs' classification and providing a comparison of several machine learning algorithms used in the referenced research. Finally, Chapter 4 outlines the state of the art and future trends of QoE estimation and management development.

2. STREAMING VIDEO DETECTION

The first step in assessing the QoE is the detection of streamed videos. This section describes approaches to detect YouTube and Netflix traffic. These approaches rely on the DNS Lookup method, as well as on fingerprinting the content based on the video streaming protocol.

2.1 DNS Lookup

As the bitrate and content specific information from encrypted traffic cannot be obtained through classic deep packet inspection techniques, the DNS Lookup method can be used to at least identify video packets, in case specific server IP addresses are known in advance. Research from [14] focuses exclusively on YouTube traffic and identifies video traffic by checking the server IP address from either the DNS response packets or TLS handshake messages. In [14], a video server IP list is built and constantly updated by checking for a specific string pattern in the DNS response or SSL/TLS handshake packets. These records are associated with a video streaming and are removed from the list if they are not hit within a week.

2.2 Fingerprinting

doi: 10.2313/NET-2018-03-1_01

Fingerprinting YouTube Traffic
The current implementation of the YouTube video service is based on two adaptive streaming modes, Apple HTTP Live Streaming (HLS) and MPEG Dynamic Adaptive Streaming over HTTP (DASH), both of which adopted SSL/TLS encrypted transmission. As a first step, the streaming mode has to be distinguished. In approaching this challenge, [14] propose a solution, which relies on the fact that each mode starts a video streaming session with the transfer of certain video index files. On one hand, the index file, DASH uses the "initsegment" file together with the related video files, which are stored on the same server. On the other hand, for HLS, the index file "manifest" and the associated video files are stored on different servers [14]. A second difference between the two streaming modes resides in the fact that, unlike DASH, HLS ClientHello messages of the SSL/TLS handshakes contain the specific string "manifest.googlevideo.co". Thirdly, the first segment after the TLS handshake differs in size, depending on the mode. Hence, in HLS, a large video block is sent right after the handshake, while in DASH, a small 1.5KB segment is sent from the video server.

Fingerprinting Netflix Traffic
The technology used by Netflix for browser-based streaming, relies on first encoding the videos as variable bitrate (VBR), followed by streaming them using DASH via Microsoft Silverlight [16]. According to [16], the combination of VBR and DASH can produce unique fingerprints for each video. Furthermore, this pattern is particularly observable for Netflix videos, as they have a higher amplitude of the bitrate variation, compared to other streaming services. The implementation proposed by [16], analyzes Netflix traffic, by leveraging two tools: adudump and OpenWPM.
The former is a program that can infer the sizes of the application data units (ADUs) transferred over each TCP connection, using the TCP sequence and acknowledgement numbers. Being able to distinguish successive video segments of an HTTPS-protected Netflix stream, adudump provides the input to the identification algorithm, as can be seen in Table 1.
OpenWPM is a framework used to conduct web measurements, being basically an automated browser, based on Firefox and Selenium, which takes a list of URLs as input and visits them sequentially.
In the first step, the video database is created using the two tools to gather the unique information for each Netflix video, by extracting fingerprints from the first 20 seconds of playback. In the second step, live traffic is logged from the network by extracting ADUs larger than 200,000 bytes, sent from a server on port 443. Finally, the live information is compared to the existing entries in the database, by performing a kd-Tree Search. Results of the success rate and detection time, obtained by [16], can be seen in Figure 1, respectively Figure 2.

2.3 Net Neutrality
While on the one hand, the goal of QoE in mobile traffic is to optimize the end user's perceived quality and increase the level of satisfaction, on the other side, traffic management measures impact other dimensions like neutrality and privacy. Not only the QoE monitoring and measurement are challenging tasks, but also the QoE control and optimization constitute a demanding duty. In order to fulfill clients'

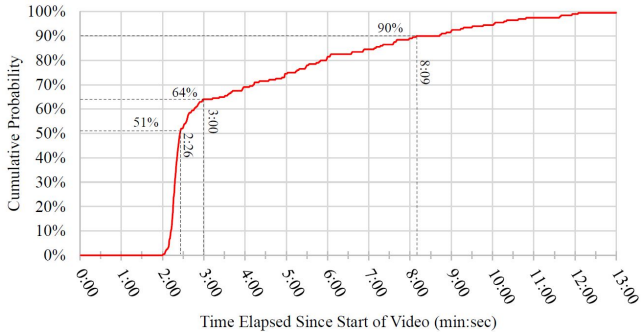


Figure 1: Cumulative probability of identifying a video before a specified amount of time has elapsed.

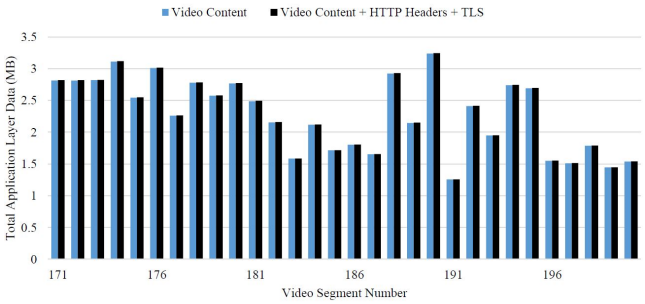


Figure 2: Netflix video overhead due to HTTP headers and TLS (Home, 3830 kbps encoding).

expectations and maintain their market shares on a competitive market, ISPs have to manage resources in a way that the QoE is not negatively impacted, while achieving cost efficiency. In a heterogeneous environment with diverse usage contexts, limited bandwidth and changing conditions, net neutrality is often affected [8].
From the technical point of view, in order to address performance issues and cope with congestions, ISPs use traffic prioritization schemes. As a result, communication services classified to be more valuable to the end user, like voice calls, are routed with higher priority than other services [8]. However, congestion issues can be solved through infrastructure updates to expand capacity.
Moreover, [10] describe also other prioritization schemes, stronger related to economic and financial arguments, rather than technical ones. Although wireless ISPs like AT&T and Verizon justify these restrictions as being related only to capacity issues, they offer monthly data plans that differentiate between the applications that are allowed to access the Internet. As a result, the tethering functionality of smartphones can be used only with a monthly data plan that allows it.
According to [6], ISPs justify using borderline practices for the traffic management for the following reasons: to ensure security, to relieve congestions, to ensure adequate QoS for selected traffic. The first reason is used when justifying blocked traffic, the second one to justify shaping on file-sharing traffic, while the third one to justify exclusive QoS for the ISPs own Voice over IP (VoIP) traffic.
In conclusion, the net neutrality issue remains an open debate point [7], as in absence of detailed regulatory policies,

ISPs continue to use the same traffic management practices, while research like [8] admittedly falls short in providing feasible solutions.

3. QOE ESTIMATION

Estimating the QoE from encrypted traffic is based on several KPIs, which constitute the ground input for the QoE estimation models. The most important KPIs [17], as well as the most common machine learning techniques are described below.

3.1 Key Performance Indicators

Initial Delay

The initial delay or start-up delay refers to the time span between a user's video request and the actual playback begin. This delay comprises two components, the network delay and the initial buffering delay. The first delay involves the required time for sending the request to the server, as well as receiving the first segments and is influenced by factors like server response times, DNS lookups and CDN redirections [3]. The second delay refers to the necessary time to fill the initial buffer required to permit a seamless playback [4].
As stated in [11] and [19], the initial delay has a small impact on the Mean Opinion Score (MOS), which is only marginally influenced by the length of the video stream, as can be seen in Figure 3.

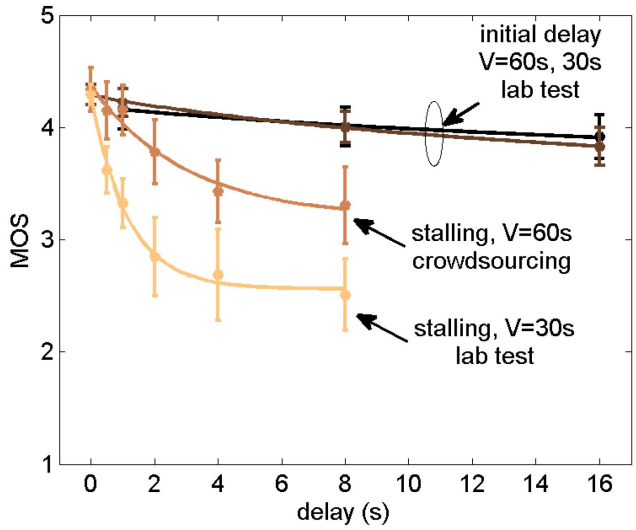


Figure 3: One stalling vs. initial delay for YouTube QoE for videos of duration V = 30 s and V = 60 s, respectively.

Stalls

A stall occurs when the content consumption rate exceeds the average download rate due to a limited network throughput. This forces the playback to freeze until the buffer is refilled with sufficient content segments. The severity of stalls is influenced by their duration and frequency during a video playback.
As reported by [12], the combination of the two factors strongly influences the MOS, being the most relevant KPI, with the highest impact on the QoE. Furthermore, as can be seen in Figure 4, sector AB of the chart is mainly a dark color region, indicating a low MOS, while the light color regions, indicating a high MOS, shown in sectors CA and BC are the result of zero packet loss rate, respectively zero packet delay.

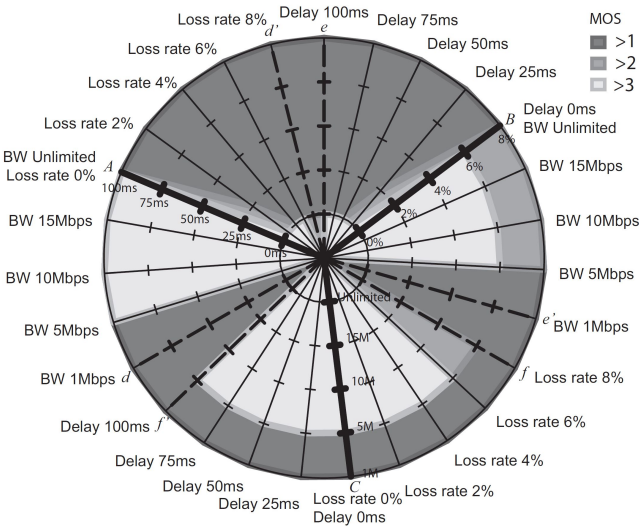


Figure 4: The radar chart mapping network QoS with QoE.

Average Representation Quality

This metric represents the average quality of all streamed video segments during a video session and is the result of the overall media throughput, measured in bits per second. However, this KPI is relevant only for video sessions based on HTTP Adaptive Streaming (HAS).
According to [9], a high average representation quality is correlated to superior user experience, as can be observed in Figure 5.

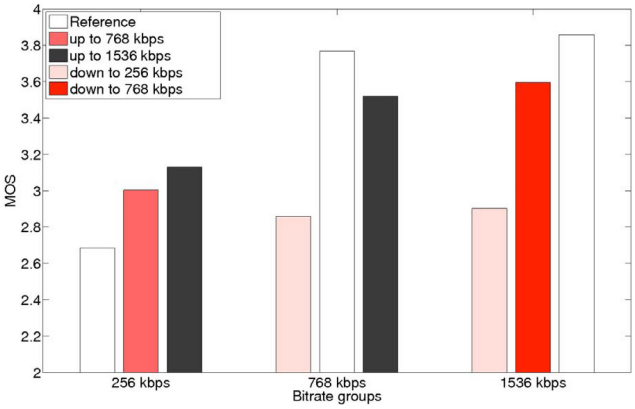


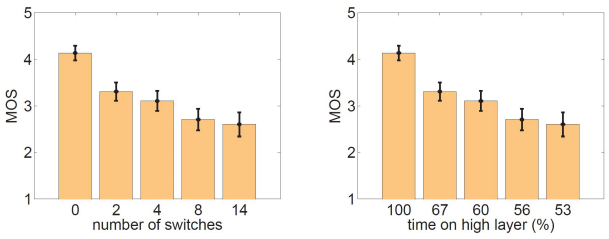
Figure 5: Bit rate switching for H.264 in WiFi.

Representation Quality Variation

Finally, the changes in the representation quality throughout a video session can significantly affect the overall QoE. This metric is based on two components, the frequency of quality changes and their amplitude. The former counts the number of quality switches during a video session and is influenced by the changing network conditions. The latter defines the

doi: 10.2313/NET-2018-03-1_01

difference in quality between two consecutive video segments in the attempt to avoid buffer depletions or to increase the video quality when the network conditions allow it.
As presented in [5], studies performed in mobile networks have shown that the representation quality variations have a high impact on the overall QoE, as can be seen in Figure 6.



(a) Number of switches (b) Time on highest layer

Figure 6: Frequency and time on highest layer changed simultaneously.

3.2 Machine Learning

Several studies propose machine learning, as a technique to quantify the correlation between QoS and QoE. Machine learning methods are used to predict the QoE based on inference rules extracted from a set of measurements reflecting the network state and the user's perception [1].
In [14], the authors propose a Machine Learning-based bitrate estimation system, which parses the bitrate information from IP packet level measurements. In order to assess the QoE based on the bitrate of video segments, a decision tree classifier is used. In the first step, YouTube traffic is identified based on the DNS Lookup method. After the HTTPS adaptive streaming protocol has been identified, the bitrate identifiers are extracted and the KPIs are computed in order to assess the vide QoE, as can be seen in Figure 7.
The solution proposed by [13] focuses on mobile devices and is based on a model built using encrypted network traffic and information collected on a client device, as illustrated in Figure 8. The process consists of three steps, data collection, data processing, and model building. Data are first collected by leveraging the YouTube Iframe API [13], which is run on the client to monitor application-level data and extract relevant KPIs. On the one hand, network traffic is captured in order to calculate traffic features like the average throughput or inter-arrival time. On the other hand, the collected application-level data, like the initial delay and stalling duration, are stored into log files, which are uploaded to the YouQ server. Based on the collected information, 33 traffic features for each video are derived to form a dataset for training the model.

Finally, a high, medium or low quality level is assigned to each video. In order to assess the quality levels, different algorithms were run, using the Waikato Environment for Knowledge Analysis (Weka), which is a suite of machine learning software written in Java.

3.2.1 *Common Machine Learning Algorithms*

This section briefly describes the principles of the most common classification algorithms used in [13] and provides an

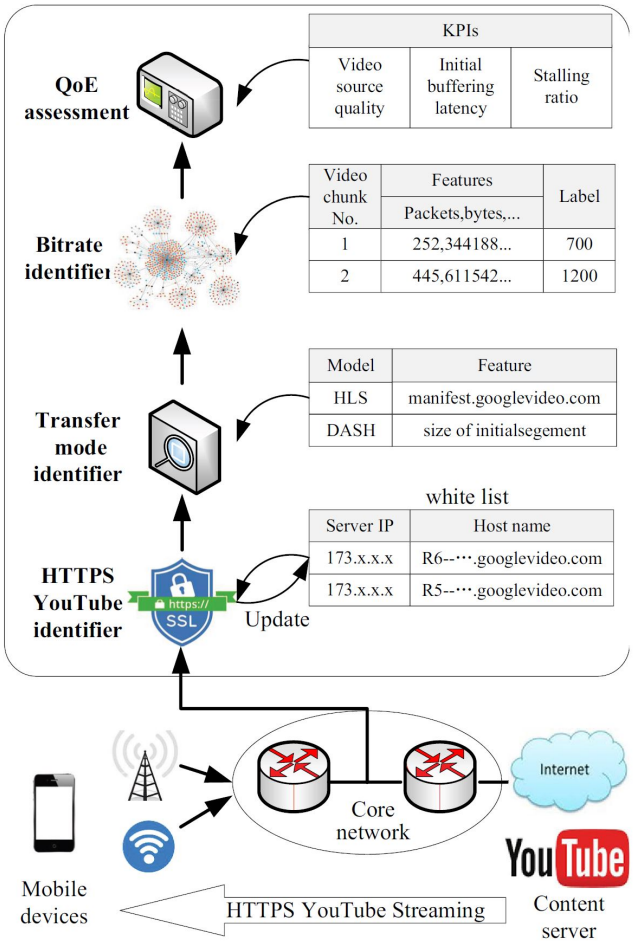


Figure 7: Architecture of QoE assessment system.

overview of the classification results.

OneR
OneR is a simple learning algorithm that creates a rule for each predictor in the data and then selects the rule with the fewest prediction errors as its single rule.

Naive Bayes
Naive Bayes is a supervised learning algorithm that uses every pair of features, as being equally relevant and statistically independent.

J48
J48 is an algorithm used to generate a decision tree, based on a top-down strategy to split the dataset on each attribute depending on its value.

SMO
The sequential minimal optimization (SMO) algorithm is based on finding a hyperplane that maximizes the minimum distance to the training set.

Random Forest
The Random Forest is a classifier that operates by constructing a multitude of decision trees and distribute instances in

doi: 10.2313/NET-2018-03-1_01

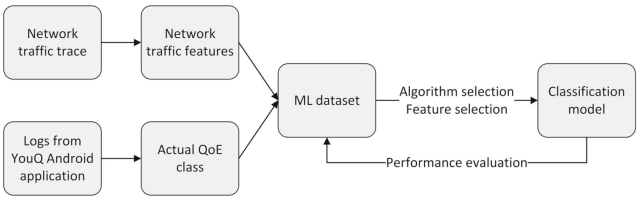


Figure 8: Approach for QoE classification based on network features.

Table 1: adudump trace of Home (3830 kbps encoding). These are segments 171-180 from Figure 2.

Timestamp	Local PC	Dir.	Netflix Server	Size (B)
1471357732.77583	134.240.17.111.31177	>	198.45.63.167.443	756
1471357736.70148	134.240.17.111.31177	<	198.45.63.167.443	2817667
1471357736.77902	134.240.17.111.31177	>	198.45.63.167.443	756
1471357740.89304	134.240.17.111.31177	<	198.45.63.167.443	2816159
1471357740.97057	134.240.17.111.31177	>	198.45.63.167.443	756
1471357744.45695	134.240.17.111.31177	<	198.45.63.167.443	2822089
1471357744.53453	134.240.17.111.31177	>	198.45.63.167.443	756
1471357748.76052	134.240.17.111.31177	<	198.45.63.167.443	3117490
1471357748.83926	134.240.17.111.31177	>	198.45.63.167.443	756
1471357752.72718	134.240.17.111.31177	<	198.45.63.167.443	2548098
1471357752.80466	134.240.17.111.31177	>	198.45.63.167.443	756
1471357756.87447	134.240.17.111.31177	<	198.45.63.167.443	3014236
1471357756.95195	134.240.17.111.31177	>	198.45.63.167.443	756
1471357760.48768	134.240.17.111.31177	<	198.45.63.167.443	2263764
1471357760.56593	134.240.17.111.31177	>	198.45.63.167.443	756
1471357764.73616	134.240.17.111.31177	<	198.45.63.167.443	2782180
1471357764.81363	134.240.17.111.31177	>	198.45.63.167.443	755
1471357768.73659	134.240.17.111.31177	<	198.45.63.167.443	2577683
1471357768.81421	134.240.17.111.31177	>	198.45.63.167.443	756
1471357772.97218	134.240.17.111.31177	<	198.45.63.167.443	2770492

the class the most decision trees agree on.

3.2.2 Classification Results

As can be seen in Figure 9, the accuracy of the selected algorithms ranges between 74.62% and 80.18% in classifying the videos according to the 3 QoE classes.

Algorithm	Selected features	Accuracy
OneR	*throughputMedian*	74.62%
Naïve Bayes	*avgPacketSize, averageInterarrivalTime, minimalInterarrivalTime, sizeThroughTimeMedian, push, interarrivalTimeThroughTimeMedian, initialThroughput2*	77.35%
SMO	*maximalSizeThroughTime, minimalInterarrivalTime, sizeThroughTimeMedian, maxThroughputThroughTime, dupack, effectiveThroguhput*	77.35%
J48	*minimalInterarrivalTime, avgInterarrivalTimeThroughTime, sizeThroughTimeStdDev, interarrivalTimeThroughTimeMedian, throughputMedian*	78.20%
Random Forest	*avgPacketSize, minimalSizeThroughTime, push, initialThroughput10, minThroughputThroughTime, interarrivalTimeThroughTimeMedian, dupack, effectiveThroughput*	80.18%

Figure 9: Classification results.

4. FUTURE TRENDS

While many studies rely on machine learning techniques to estimate the QoE, most of them use only a limited number of influence factors, while few approaches map simultaneously the impact of multiple factors to obtain a multidimensional QoE model [13].

Future work also focuses on developing the existing platforms in order to increase the degree of support for different devices, operating systems, network types and other environment variables [15] .

5. CONCLUSION

While the field of online content distribution has been continuously gaining momentum, the importance of good QoE has significantly increased. In order to remain competitive, ISPs need to adapt their QoE detection techniques for encrypted traffic, as the big OTT providers of video streaming services already use end-to-end encryption to a big extent. Existing studies approach this challenge relying on the current video streaming protocols and ML techniques, but are limited to specific platforms, devices, operating systems and types of networks. While the presented ML methods show promising results, there is still room for improvements in the areas of performance and accuracy.
On the other side, current work shows that the legislation lacks consistency in regulating the QoE and traffic management to preserve principles of the net neutrality.

6. REFERENCES

[1] S. Aroussi and A. Mellouk. Survey on machine learning-based qoe-qos correlation models. In *2014 International Conference on Computing, Management and Telecommunications (ComManTel)*, pages 200–204, 2014.

[2] Cisco Visual Networking Index: Forecast and Methodology, 2016-2021. June 6, 2017. `https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf`.

[3] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, and K. Papagiannaki. Measuring video qoe from encrypted traffic. In *Proceedings of the 2016 Internet Measurement Conference*, pages 513–526, 2016.

[4] T. Hossfeld, S. Egger, R. Schatz, M. Fiedler, K. Masuch, and C. Lorentzen. Initial delay vs. interruptions: Between the devil and the deep blue sea. In *2012 Fourth International Workshop on Quality of Multimedia Experience*, pages 1–6, 2012.

[5] T. Hossfeld, M. Seufert, C. Sieber, and T. Zinner. Assessing effect sizes of influence factors towards a qoe model for http adaptive streaming. In *2014 Sixth International Workshop on Quality of Multimedia Experience (QoMEX)*, pages 111–116, 2014.

[6] S. Jordan. Some traffic management practices are unreasonable. In *2009 Proceedings of 18th International Conference on Computer Communications and Networks*, pages 1–6, Aug.

[7] S. Jordan. Four questions that determine whether traffic management is reasonable. In *2009 IFIP/IEEE International Symposium on Integrated Network Management*, pages 137–140, June 2009.

[8] S. Jordan. Traffic management and net neutrality in wireless networks. *IEEE Transactions on Network and Service Management*, 8(4):297–309, December 2011.

[9] B. Lewcio, B. Belmudez, A. Mehmood, M. WÃd'ltermann, and S. MÃűller. Video quality in next generation mobile networks x2014; perception of time-varying transmission. In *2011 IEEE International*

Workshop Technical Committee on Communications Quality and Reliability (CQR), pages 1–6, 2011.

[10] L. Martinez, O. S. J. Alvarez, and J. Markendahl. Net neutrality principles and its impact on quality of experience based service differentiation in mobile networks. In *2015 Regional Conference of the International Telecommunications Society (ITS)*, 2015.

[11] R. K. Mok, E. W. Chan, X. Luo, and R. K. Chang. Inferring the qoe of http video streaming from user-viewing activities. In *Proceedings of the First ACM SIGCOMM Workshop on Measurements Up the Stack*, pages 31–36, 2011.

[12] R. K. P. Mok, E. W. W. Chan, and R. K. C. Chang. Measuring the quality of experience of http video streaming. In *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*, pages 485–492, 2011.

[13] I. Orsolic, D. Pevec, M. Suznjevic, and L. Skorin-Kapov. A machine learning approach to classifying youtube qoe based on encrypted network traffic. *Multimedia Tools and Applications*, pages 1–35, 2017.

[14] W. Pan, G. Cheng, H. Wu, and Y. Tang. Towards qoe assessment of encrypted youtube adaptive video streaming in mobile networks. In *2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS)*, pages 1–6, 2016.

[15] N. Radics, P. SzilÃągyi, and C. VulkÃąn. Insight based dynamic qoe management in lte. In *2015 IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1746–1751, 2015.

[16] A. Reed and M. Kranch. Identifying https-protected netflix videos in real-time. In *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*, pages 361–368, 2017.

[17] Subjective audiovisual quality assessment methods for multimedia applications. December 3, 1998. `http://handle.itu.int/11.1002/1000/4538`.

[18] YouTube's road to HTTPS. August 1, 2016. `https://youtube-eng.googleblog.com/2016/08/youtubes-road-to-https.html`.

[19] G. Zhai, J. Cai, W. Lin, X. Yang, W. Zhang, and M. Etoh. Cross-dimensional perceptual quality assessment for low bit-rate videos. *IEEE Transactions on Multimedia*, 10(7):1316–1324, Nov 2008.

Comparison of IoT Data Protocol Overhead

Vasil Sarafov
Advisor: M.Sc. Jan Seeger
Seminar Future Internet SS2017
Chair of Network Architectures and Services
Departments of Informatics, Technical University of Munich
Email: sarafov@cs.tum.edu

ABSTRACT

The Internet of Things is expanding at fast pace and every year constrained devices that rely on intercommunication are being deployed. Knowing how much overhead a communication mechanism adds to the system can be of huge importance for its optimal utilization and prevent performance degradation. In this paper we construct an abstract theoretical model for deriving and comparing the overhead of the WebSocket, CoAP and MQTT protocols when sending upstream an arbitrary number of data packets. We then validate the end results with an experiment and show that CoAP with non-confirmable messages demonstrates the least overhead when no datagrams are lost, followed by MQTT with QoS 0, which outperforms the WebSocket protocol by a tiny margin.

Keywords

Protocol overhead, Throughput, Performance Comparison, CoAP, MQTT, WebSocket, IoT, Internet of Things, Data Protocols

1. INTRODUCTION

For decades people have been involved in a technological revolution, which has opened a new chapter in human history. Starting from science and education and continuing with important industrial and medical applications, the Internet has been accelerating the world's progress for years. Even more, it has now become one of the most important communication media.

The tremendous advance in the fields of electronics, robotics and artificial intelligence has led to the next stage of this revolution - bringing the *Internet* to a new state where it can be used for interconnecting *Things* and *Machines* that are not operated by people and communicate autonomously with one another. This phase was given the name *The Internet of Things* (IoT).

One of the core concepts of IoT is exactly the communication between interconnected devices. In most cases the connected nodes are operating in constrained environments and have very limited resources such as CPU power, RAM and available energy. Therefore an optimal communication mechanism is a must. Calculating the application protocol overhead is a step towards such optimization and in this paper we compare the overhead of three data protocols that are widely used in various IoT services - WebSocket, Constrained Application Protocol (CoAP) and MQTT.

The WebSocket Protocol was designed as a solution to the problem which the Web was facing for many years - the lack of full duplex asynchronous communication between a client and a server. Since WebSocket allows that and is fairly easy to integrate in existing HTTP infrastructure, many IoT devices and platforms choose to implement it.

CoAP's aim is to be for the IoT world what HTTP is for the Web. It brings the well known REST model [8] to networks with constrained nodes by relying on UDP and very small protocol overhead.

MQTT is a protocol that follows the publish-subscribe communication pattern and provides a very convenient decoupling of the peers in large distributed systems that can be used to optimize complex business logic. It was designed to be light, highly scalable and easy to adjust and implement on the client side, which makes it a perfect fit for the IoT domain.

The paper is structured as follows. In Chapter 2 we present a very simple mathematical model that is used in Chapter 3 to compare WebSocket, CoAP and MQTT, where the protocols are described in a more detailed manner as well. We validate our theoretical estimations with an experiment in Chapter 4 and make the overall conclusion in Chapter 6. In addition, we explain how the proposed model can be extended so that it can be utilized for estimating the protocol overhead in more complex real-world applications.

Related Work

Existing work compares different lightweight internet protocols mainly based on the provided feature set. Performance evaluation is usually done only empirically and for a very specific use case. In this paper we provide a general theoretical overhead comparison that is in addition experimentally validated. Its results can be applied for an arbitrary scenario, independently of the physical connection.

In [20] Thangavel et al design and implement a middleware for wireless sensor networks. By using the common middleware they empirically evaluate the performance of CoAP and MQTT. [12] compares CoAP's and MQTT's performance in NBIoT networks, emphasizing the adaptation of both data protocols in regard to the limitations of the NBIoT physical properties. [7] provides a qualitative and quantitative comparison between MQTT and CoAP for smartphone-based sensor systems. The quantitative analysis is executed in a

doi: 10.2313/NET-2018-03-1_02

WiFi-based network for a publish/subscribe use case and therefore utilizes CoAP with its observe extension. Very similarly, [2] discusses the feature differences between the same two protocols and benchmarks them based on performance criteria such as overhead and energy consumption. In [21] Yokotami and Sasaki measure the performance differences between HTTP and MQTT in a publish/subscribe scenario. Their comparison is based on bandwidth and server resource allocation costs for up to 1000 connected IoT devices.

A very detailed performance examination of CoAP, MQTT and DDS (Data Distribution Service for real-time systems) is presented in [5] by Chen and Kunz. They quantitatively compare the protocols' bandwidth consumption, experienced latency, packet loss and operation in low quality wireless networks in terms of a medical IoT use case.

2. MATHEMATICAL MODEL

In this section we present a very short and simplified mathematical model for deriving the overhead of an arbitrary data communication protocol in an abstract form. Our goal is to apply it directly to the protocols described in Chapter 3 and compare the end results.

Considering the fact that our aim is to compare the overhead of data communication protocols for the *IoT application domain*, we make the following observations and assumptions, which will help us simplify our model:

- We are interested only in the size overhead of the protocol (how much additional control information is needed to send x amount of application data). This is of huge importance for constrained devices, such as IoT nodes, because every additional byte being processed implies a higher energy consumption.
- Latency is not taken into consideration since it is strongly dependent on the physical properties of the underlying network. We focus on *application data protocols* which can be utilized in networks with different physical attributes.
- Following the OSI layering model [4], data protocols have the same overhead for the Physical (L1), Data Link (L2) and Network Layers (L3) when they are used in the same environment. Therefore we are interested in comparing the overhead differences starting with the Transport Layer (L4).
- For the sake of simplicity we assume that *no* IP fragmentation is taking place which holds true in many constrained networks [19, Section 4.6]. This assumption implies a payload size of less than 1024 Bytes for the data protocols that were chosen, which is further justified in Chapter 3.2.
- Whether or not the application data is compressed is completely irrelevant to the model. This is because the payload is represented by its total size in the model calculations.
- A typical IoT use case is when a device sends data (e.g. sensor values) to a server or a gateway in a particular time slot (also known as a communication slot) after it has established connection. In the remaining time it would normally process time critical tasks or simply enter a deep sleep mode to save energy. For the sake of simplicity but still evaluating a realistic use case, we restrict ourselves only to upstream communication and examine each protocol in a scenario where the client (an IoT device) is the data source and the server is the data collector.

Parameter	Description	Constraints
p	Sum of the size of the headers from Layer 1, 2 and 3 that are present in every sent packet.	$p \in \mathbb{N}$
b	Indicates whether the data protocol has an opening and closing handshake ($b = 1$) or not ($b = 0$).	$b \in \{0, 1\}$
H_o	Size of the opening handshake in Bytes (Layer 4 upwards).	$H_o \in \mathbb{N}_0$
H_c	Size of the closing handshake in Bytes (Layer 4 upwards).	$H_c \in \mathbb{N}_0$
n	Number of the available communication slots until the connection is closed.	$n \in \mathbb{N}$
x_i	Size of the application data in Bytes that is to be sent in communication slot i.	$x_i \in \mathbb{N}_0, \leq 1024$
h_i	Size of the data protocol header in Bytes that is needed to wrap x_i.	$h_i \in \mathbb{N}$

Table 1: Description of the parameters used in the mathematical model for evaluating the overhead of an arbitrary data protocol.

In Table 1 are presented the parameters which describe our abstract data protocol model.

An important observation is that for CoAP [19, Section 3] and MQTT [1, Section 3.3] the headers that wrap the application data have a constant size. Hence for both protocols holds:

$$h_i = h_j, \forall i, j \in \{1, 2...n\}$$

For the WebSocket protocol the headers that wrap the application data differ with at most 2 Bytes in size [13, Section 5.2]. Therefore for WebSocket holds:

$$|h_i - h_j| \leq 2, \forall i, j \in \{1, 2...n\}$$

Considering the above observations and for simplification reasons, with negligible error we assume a constant header size:

$$h := h_i = h_j, \forall i, j \in \{1, 2...n\}$$

Furthermore, we define D_{app} as the total amount of appli-

doi: 10.2313/NET-2018-03-1_02

cation data that will be sent in a single connection:

$$D_{app} := \sum_{i=1}^{n} x_i = n\widetilde{x} \tag{1}$$

where $\widetilde{x}$ is the average of all x_i.

Analogously we obtain the total amount of data (including all L1-L7 overheads) that is sent when D_{app} is dispatched as D_{total}:

$$\begin{aligned} D_{total} := & \underbrace{b(c_1 p + H_o)}_{\text{Opening Handshake}} + \underbrace{\sum_{i=1}^{n}(p + h_i + x_i)}_{\text{Sending of } D_{app}} + \underbrace{b(c_2 p + H_c)}_{\text{Closing Handshake}} \\ & = \underbrace{bp(c_1 + c_2) + np}_{\text{L1-L3 overhead}} + \underbrace{bH_o + bH_c + nh}_{\text{L4-L7 overhead}} + \underbrace{n\widetilde{x}}_{D_{app}} \end{aligned} \tag{2}$$

where c_1 and c_2 are the number of packets that are needed to complete the opening and closing handshake respectively.

Hence the actual data protocol overhead for n communication slots is given by the overhead sum between Layer 4 and Layer 7:

$$\omega(n) := bH_o + bH_c + nh \tag{3}$$

Furthermore, for the protocol's throughput τ we obtain:

$$\begin{aligned} \tau(n, \widetilde{x}) &= \frac{D_{app}}{D_{total}} \\ &= \frac{n\widetilde{x}}{\omega(n) + n\widetilde{x} + bp(c_1 + c_2) + np} \end{aligned} \tag{4}$$

Obviously $\forall n, x \in \mathbb{N}.\tau(n, \widetilde{x}) < 1$ and $\tau(n, \widetilde{x})$ is a strongly monotone increasing function. We use this observation in Chapter 4.

3. THEORETICAL PROTOCOL OVERHEAD ESTIMATION

In this section we apply the mathematical model described in Chapter 2 for the WebSocket, CoAP and MQTT protocols respectively. We do this by giving the exact Layer 4 to Layer 7 costs and approximating where necessary. Encryption layering with Transport Layer Security (TLS) [15] for WebSocket and MQTT and Datagram Transport Layer Security (DTLS) [16] for CoAP is omitted for the sake of simplicity.

Furthermore proxy and cache optimizations, which can in many cases boost the performance of the communication system, are also not taken into consideration. The reason for this assumption is that proxy usage is very often tightly coupled to the application's logic and therefore is not suitable for our abstract evaluation framework.

3.1 The WebSocket Protocol

WebSocket is a data protocol that exposes TCP on a higher abstraction level so that it can be used almost directly by Web browser applications. It was standardized by the IETF in 2011 with RFC 6455 [13].

The motivation behind its design was to solve the lack of asynchronous communication from server to client in HTTP and provide a long living full duplex connection that can be integrated in existing HTTP infrastructure [18], making it suitable for IoT use cases where a real time bidirectional interaction with the device is desired. The compatibility in terms of deployment between the two completely different protocols is achieved via the HTTP upgrade header which notifies the server to change the protocol from HTTP to WebSocket.

A WebSocket-based communication stack is layered on top of TCP. Therefore the client and server should not take care of data fragmentation or packet acknowledgement. Similarly to HTTP, a WebSocket connection can be secured with TLS. In most cases the real application protocol (defined as subprotocol in [13, Section 1.9]) is layered directly over the WebSocket, which means that WebSocket is used only for the connection and does not create any constraints to the business logic of the system.

The life cycle of a non-TLS secured WebSocket connection is shown on Figure 1. It consists of:

- A 3-way TCP opening handshake that costs approximately 60 Bytes[1].
- A 2-way WebSocket opening handshake. Its size can vary based on the connection meta information such as the endpoint and server hostname [13, Section 1.3]. A good approximation is $\approx$ 310 Bytes (Client side $\approx$ 170 Bytes, Server side $\approx$ 140 Bytes).
- A routine for sending the actual application data over WebSocket. Each packet that is not buffered by the sender (i.e it is sent directly) is wrapped in a frame of total size 12 or 14 Bytes. Frames are used instead of a direct streaming approach in order to prevent mandatory buffering and to allow dynamically adjustable multiplexing of the duplex communication in future versions of the protocol [13, Section 5.2]. Furthermore, very often resource-constrained IoT devices cannot support a buffering mechanism because of lack of enough memory.
- A 2-way WebSocket closing handshake, which consists of two empty frames, 14 Bytes each [13, Section 1.4]. Thus it has a total size of 28 Bytes.
- A 3-way TCP closing handshake that costs approximately 60 Bytes.

Hence we obtain together with the TCP message and acknowledgement wrapping of the WebSocket messages the approximated values for our overhead estimation model in Table 2.

From that we directly derive the final representation of the WebSocket data protocol overhead, using Equation 3:

$$\omega_{WebSocket}(n) \approx 600 + 54n \tag{5}$$

[1]We assume that the TCP header size is 20 Bytes (TCP header options are not considered)

doi: 10.2313/NET-2018-03-1_02

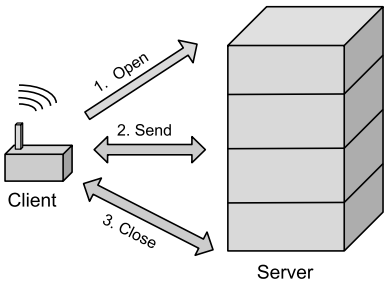


Figure 1: Life cycle of a bidirectional WebSocket connection

Parameter	Value
b	1
H_o	≈ 430
H_c	168
h	54

Table 2: Model values for the WebSocket protocol

3.2 Constrained Application Protocol

The Constrained Application Protocol (CoAP) is a web transfer protocol that follows the request-response communication pattern (Figure 2) and is intended for use with constrained nodes and networks in machine-to-machine (M2M) applications. The protocol was officially standardized by the IETF in June 2014 with RFC 7257 [19].

CoAP's general purpose is to allow a subset of the convenient REST architecture that HTTP provides for the Web [8] to be utilized by applications running on microcontrollers. Typically, microcontrollers have limited hardware resources such as memory, CPU power and energy and often communicate through constrained networks such as 6LoWPAN [14]. This makes the direct utilization of REST with the full HTTP stack hard. Nevertheless, CoAP can be further extended to support the observe communication pattern by executing augmented GET requests [9].

To provide flexibility and address the constrained nature of the nodes, CoAP is layered over UDP. Using TCP instead of UDP is theoretically possible but not standardized. Additional layering over DTLS secures the connection [16].

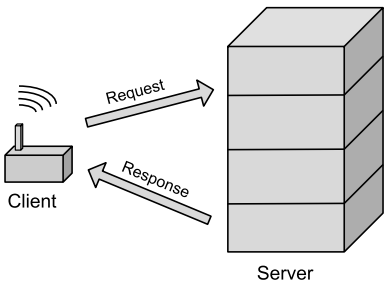


Figure 2: The CoAP client requests resources from the server by specifying a Request Verb and an endpoint. The server responds accordingly with the requested resource.

Parameter	Value	Note
b	0	No handshake
H_o	0	No handshake
H_c	0	No handshake
h	38	non-confirmable request with non-confirmable response
h	50	non-confirmable request with confirmable response
h	38	confirmable request with piggybacked response
h	50	confirmable request with separate non-confirmable response
h	62	confirmable request, separate confirmable response

Table 3: Model values for the CoAP protocol and all request/response message types (upstream communication only)

CoAP deals with the lack of order and unreliable transmission of UDP datagrams by utilizing a very simple messaging layer over the actual request/response. Thus important and time-critical packets can be acknowledged which is a very simple form of an adjustable quality of service (QoS) [19, Section 4]. A CoAP request can be in either confirmable or in non-confirmable message. In case of a confirmable request, the server must send an acknowledgement message. This message can either contain the response (called piggybacked response) or be empty (the response is sent separately). Following the same strategy, separate responses can be wrapped inside either confirmable or non-confirmable messages [19, Section 4.2].

CoAP is intended to be used without IP fragmentation. Hence problems can occur when larger amounts of data need to be transferred (example: Over-the-Air firmware updates). Therefore RFC 7959 proposes a blockwise transfer technique for CoAP messages [3]. However, a payload size of 1024 Bytes can be assumed to be transported without any fragmentation through a normal not constrained network [19, Section 4.6].

The CoAP request/response header has a fixed size of 11 Bytes when cache is disabled. For our estimation we use a 4 Bytes long token to match requests with responses which is the half of the allowed length [19, Section 3]. An acknowledgement message costs exactly 4 Bytes and the underlying UDP header is 8 Bytes long. With this information and the assumption that the application data is transferred by the client in an upstream manner (for example via POST requests) and the server's response does not contain any payload but only control information such as status code, we obtain the protocol overhead for all 5 different possibilities of the request/response life cycle (Table 3).

Thus we can conclude:

$$38n \leq \omega_{CoAP}(n) \leq 62n \tag{6}$$

doi: 10.2313/NET-2018-03-1_02

3.3 MQTT

The MQTT protocol is a client-server publish-subscribe messaging transport protocol. It was created in 1999 because of a use case where a protocol for minimal battery loss and minimal bandwidth connecting oil pipelines over satellite connection was needed [10]. The current version is 3.1.1, which as of 2014 is an OASIS and as of 2016 an ISO standard [1, 11].

MQTT is designed with simplicity in mind. It is lightweight and similarly to CoAP is suitable for machine-to-machine communication in constrained environments, where the code footprint and network bandwidth are scarce. It is layered over TCP and can be secured with TLS. In general, MQTT can function over an arbitrary transport protocol that provides ordered, lossless bi-directional connections. Hence it can be even layered on top of The WebSocket Protocol (Chapter 3.1), which makes possible writing browser-based MQTT clients.

Although the protocol is TCP based, it provides three different levels for Quality of Service (QoS) for the message delivery [1, Section 4.3]:

- QoS 0 where messages are assured to arrive at most once, hence can be lost when connection problems occur. In most cases QoS 0 is enough since MQTT can take advantage of TCP's connection reliability mechanisms.
- QoS 1 where messages are assured to arrive but duplicates can occur. This level requires a 2-way handshake for each sent message.
- QoS 2 where messages are assured to arrive exactly once. This level and QoS 1 are intended for systems where TCP's mechanisms are not enough - for example transaction systems or services that are unreliably interconnected on the physical level such as satellites. QoS 2 requires a 4-way handshake for each sent message.

QoS is important because it allows controlling the protocol overhead and thus the network can manage its quality alone when bandwidth or connection problems occur. Furthermore, QoS levels 1 and 2 imply a session which makes MQTT stateful when needed.

An MQTT connection is *exactly one* of the following two types:

1. Publishing Connection - A client connects to the server (also known as message broker) to publish multiple messages during the lifetime of the connection. (Figure 3).
2. Subscription Connection - A client connects to the broker and subscribes to message topics. During the lifetime of the connection it receives the publications of other clients which are forwarded by the broker.

Generally speaking, the message broker acts as an intermediary between the publishers and subscribers and makes

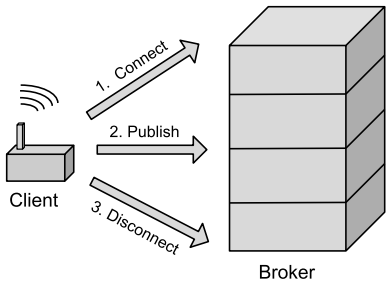


Figure 3: Connection life cycle of a client publishing information to an MQTT broker

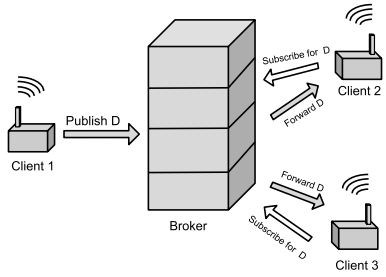


Figure 4: MQTT based publish-subscribe communication between peers

sure that the messages are delivered, based on the QoS (Figure 4). Without loss of generality, we evaluate the *publishing connection* since the *subscription connection* is identical and provides exactly the same overhead. Moreover, we are not taking into consideration the in-built username/password mechanism that MQTT provides because TLS layering is omitted. From a security point of view sending the authentication credentials in plain text makes no sense and in Chapters 3.1 and 3.2 we have evaluated the WebSocket and CoAP protocols respectively without any authentication mechanism.

We take the TCP overhead into consideration as described in Chapter 3.1. The MQTT fixed header is exactly 2 Bytes long [1, Section 2.2] and considering the message exchange algorithms described in [1, Section 3], we present in Table 4 the values for our overhead estimation model. With that said we conclude the lower (QoS 0) and upper (QoS 2)

Parameter	Value	Note
b	1	Handshake present
H_o	≈ 200	TCP and MQTT opening handshake with Will message
H_c	102	TCP and MQTT closing handshake
h	42	QoS 0, no authentication
h	86	QoS 1, no authentication
h	174	QoS 2, no authentication

Table 4: Model values for the MQTT protocol for all QoS message types (publishing connection only)

doi: 10.2313/NET-2018-03-1_02

bounds for the overhead of an MQTT-based communication:

$$300 + 42n \leq \omega_{MQTT}(n) \leq 300 + 174n \quad (7)$$

4. EXPERIMENTAL PROTOCOL OVERHEAD COMPARISON

We now present an empirical validation of the protocol overhead estimations that were made in Chapter 3.

4.1 Structure of the Experiment

The experiment was conducted in a local WiFi network with IPv4-based addressing and the following hardware:

- Client - Raspberry PI model B (Quad Core 1.2GHz Broadcom BCM283, 1GB RAM) running Ubuntu Core Linux 16 with kernel version 4.4.0-72-generic.
- Server - Laptop with Quad Core CPU Intel i7-2620M 3.4GHz, 8GB RAM, running Fedora Linux with kernel version 4.12.14-300.fc26.x86_64.

The total amount of bytes for each execution was measured with Wireshark [6]. When needed, packet loss was simulated on the client side using the Linux kernel module netem[2]. Further information about the used software components and the code for the clients and servers can be found in Appendix A.

4.2 Experiment Results

We use the throughput equation (Equation 4) we derived in Chapter 2 to compare the protocols. Moreover we also know from the same equation that the following statement holds for arbitrary protocols A and B:

$$\omega_A(n) < \omega_B(n) \Rightarrow \tau_A(n, \widetilde{x}) > \tau_B(n, \widetilde{x})$$

Thus from equations 5, 6 and 7 by comparing the estimated constants, we expect that CoAP with non-confirmable requests and responses will perform at best, then MQTT with QoS 0 and the WebSocket protocol will share the second and third place.

On Figure 5 we can see the experiment results for up to 100 communication slots, i.e. up to 100 data packets were sent through the lifetime of a single connection, and average application data size of 128 Bytes with no packet loss. For the MQTT and CoAP protocols we have used those protocol configurations, which showed the lower and upper overhead bounds respectively according to equations 6 and 7. Those are QoS 0 and QoS 2 for MQTT (Chapter 3.3) and non-confirmable requests/responses and confirmable requests with separate confirmable responses for CoAP (Chapter 3.2).

On Figure 6 we can see the results for the same experiment configuration but with a simulated 20% packet loss on the client side. Important to note is that packet losses are independently distributed, i.e. networking burst was not considered. Furthermore, non-confirmable CoAP messages are not retransmitted and thus completely lost when datagrams are dropped. This explains the unusually higher throughput values for the non-confirmable CoAP scenario.

Figure 5: $\tau(n, \widetilde{x})$ for $\widetilde{x} = 128$ and no packet loss

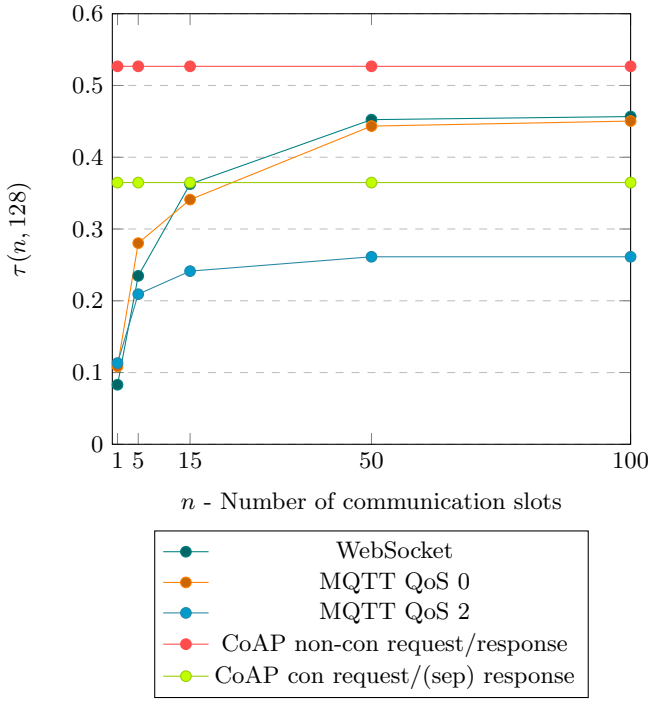
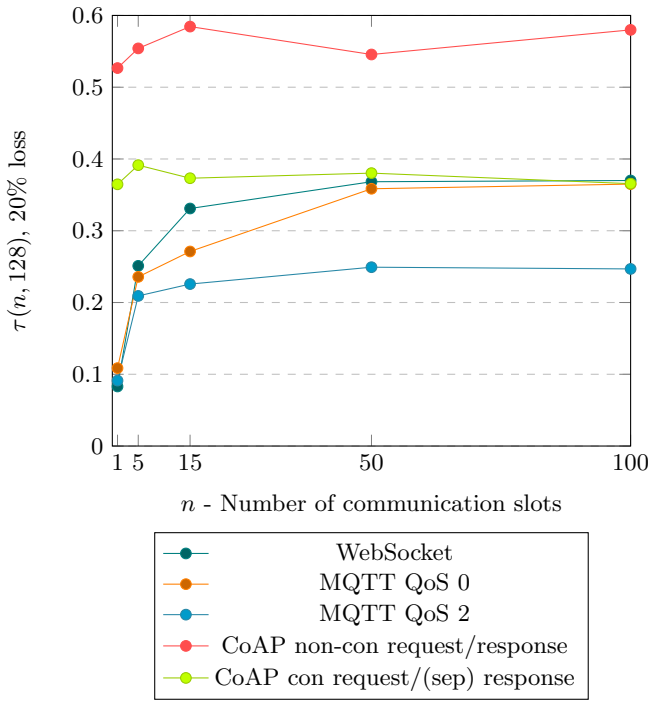


Figure 6: $\tau(n, \widetilde{x})$ for $\widetilde{x} = 128$ with 20% packet loss



doi: 10.2313/NET-2018-03-1_02

As expected, CoAP has a constant throughput when no datagrams are lost since it lacks an opening/closing handshakes. On the other hand, MQTT and WebSocket start with a poor throughput because of the handshake costs but outperform CoAP with confirmable requests and separate confirmable responses after approximately 15 communication slots (Figure 5). However, this tendency does not hold true when packets are lost. In that case CoAP with confirmable messages outperforms both TCP data protocols although all three of them converge to the same throughput rate for larger number of communication slots (Figure 6). In the general case, MQTT with QoS 2 performs at worst because of the large number of control messages that are exchanged between the client and the server. As a second result of that its overall throughput grows much slower than the one of its QoS 0 companion but remains almost the same when packets are lost.

The experiment was executed two more times with the same amounts of communication slots and payload sizes of 64 and 512 Bytes respectively, without packet loss. An important observation is that MQTT QoS 0 outperforms WebSocket faster with bigger payloads and all protocols achieve a more than two times better throughput when the average application data size is increased from 64 to 512 Bytes. In the latter case CoAP's best throughput jumps from 38% to 83%. MQTT with QoS 0 improves its throughput from 29% to 77%.

As it can be observed in all cases of the experiment execution, our theoretical estimations in Chapter 3 are correct and CoAP with non-confirmable requests/responses demonstrates the best throughput, followed by WebSocket and MQTT with QoS 0, which share the second place.

5. FURTHER WORK

Important assumptions were made in Chapter 3 in order to simplify the theoretical overhead estimation model. Not considering TLS securing for WebSocket and MQTT and DTLS securing for CoAP is a huge drawback, although the same results as in Chpater 4.2 are expected. Additionally taking into account cache and proxy optimization best-practices will surely complicate the model but also allow its direct utilization in real-world applications.

An important application layer protocol for the IoT domain which we did not discuss is the Extensible Messaging and Presence Protocol (XMPP) [17]. Comparing the many different IoT extensions for XMPP to CoAP and MQTT will be of a huge benefit, considering the fact that XMPP is heavily used in IoT platforms and services.

6. CONCLUSION

Knowing how much overhead a data protocol generates is of huge importance in cases where the networking nodes and the network itself are resource-constrained. IoT devices face many hardware limitations and as a result of that they should implement suitable communication protocols.

We showed that the theoretical overhead of CoAP with non-confirmable requests and responses is the least when compared to MQTT and WebSocket for the same amount of requests and data sent upstream, using an identical physical connection layer. Utilizing CoAP in this configuration, however, comes with the disadvantage of a higher probability of loosing packets. Furthermore, CoAP provides a constant throughput when no datagrams are lost, which does not always hold true in constrained networks.

MQTT with QoS 0 demonstrates the second best overhead. Compared against CoAP, it relies on TCP and hence can handle lost packets on the Transport Layer. Moreover, MQTT QoS 0 has a better throughput than CoAP with confirmable requests and separate confirmable responses, when no packet loss occurs. Otherwise CoAP's reliable configuration performs better because of the lighter UDP overhead. Using MQTT QoS 2 on the other hand generates too much overhead and causes the protocol to perform at worst. This is albeit understandable, considering the fact that MQTT QoS 2 was designed for satellite networks where TCP is unreliable.

WebSocket demonstrates almost the same throughput as MQTT with QoS 0. What both of these protocols share in common is the additional cost that the peers must pay to open the connection. As the connection is kept alive and data is being sent, the initialization costs start paying off and the maximum achievable throughput is reached.

All three protocols increase their throughput when more application data is sent in a single dispatch. Rising the average data packet size from 64 to 512 Bytes improves the overall throughput more than two times.

The above results compare the three protocols *only based on their overhead*. This, however, is not applicable for use cases where the application's business logic is tightly coupled to a very specific type of connection. CoAP provides a request/response mechanism which makes it suitable for general use cases that do not rely on a long living connection. On the other hand achieving a full duplex asynchronous communication between the client and the server with CoAP is impossible. If this is desired, WebSocket is to be considered. When an efficient mass distribution of data to interested parties in the system is wanted, MQTT should be chosen.

7. REFERENCES

[1] MQTT Version 3.1.1 Plus Errata 01, Dec. 2010.

[2] S. Bandyopadhyay and A. Bhattacharyya. Lightweight internet protocols for web enablement of sensors using constrained gateway devices. In *Computing, Networking and Communications (ICNC), 2013 International Conference on*, pages 334–340. IEEE, 2013.

[3] C. Bormann and Z. Shelby. Block-Wise Transfers in the Constrained Application Protocol (CoAP). RFC 7959, Aug. 2016.

[4] N. Briscoe. Understanding the OSI 7-layer model. *PC Network Advisor*, 120(2), 2000.

[5] Y. Chen and T. Kunz. Performance evaluation of iot protocols under a constrained wireless access network. In *Selected Topics in Mobile & Wireless Networking*

[2]https://wiki.linuxfoundation.org/networking/netem

(MoWNeT), 2016 International Conference on, pages 1–7. IEEE, 2016.

[6] G. Combs et al. Wireshark. *Web page: http://www. wireshark. org/last modified*, pages 12–02, 2007.

[7] N. De Caro, W. Colitti, K. Steenhaut, G. Mangino, and G. Reali. Comparison of two lightweight protocols for smartphone-based sensing. In *Communications and Vehicular Technology in the Benelux (SCVT), 2013 IEEE 20th Symposium on*, pages 1–6. IEEE, 2013.

[8] R. T. Fielding and R. N. Taylor. *Architectural styles and the design of network-based software architectures*. University of California, Irvine Doctoral dissertation, 2000.

[9] K. Hartke. Observing Resources in the Constrained Application Protocol (CoAP). RFC 7641, Sept. 2015.

[10] E. B. HiveMQ. MQTT Essentials: Part 1 Introducing MQTT. http://www.hivemq.com/blog/mqtt-essentials-part-1-introducing-mqtt, 2015.

[11] Information technology – Message Queuing Telemetry Transport (MQTT) v3.1.1. Standard, International Organization for Standardization, Geneva, CH, June 2016.

[12] T. Maksymyuk, M. Brych, S. Dumych, and H. Al-Zayadi. Comparison of the iot transport protocols performance over narrowband-iot networks. *Internet of Things and Ubiquitous Communications*, 1(1):25–28, 2017.

[13] A. Melnikov and I. Fette. The WebSocket Protocol. RFC 6455, Dec. 2011.

[14] G. Montenegro, J. Hui, D. Culler, and N. Kushalnagar. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944, Sept. 2007.

[15] E. Rescorla and T. Dierks. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, Aug. 2008.

[16] E. Rescorla and N. Modadugu. Datagram Transport Layer Security Version 1.2. RFC 6347, Jan. 2012.

[17] P. Saint-Andre. Extensible Messaging and Presence Protocol (XMPP): Core. RFC 6120, Mar. 2011.

[18] P. Saint-Andre, S. Loreto, S. Salsano, and G. Wilkins. Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP. RFC 6202, Apr. 2011.

[19] Z. Shelby, K. Hartke, and C. Bormann. The Constrained Application Protocol (CoAP). RFC 7252, June 2014.

[20] D. Thangavel, X. Ma, A. Valera, H.-X. Tan, and C. K.-Y. Tan. Performance evaluation of mqtt and coap via a common middleware. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2014 IEEE Ninth International Conference on*, pages 1–6. IEEE, 2014.

[21] T. Yokotani and Y. Sasaki. Comparison with http and mqtt on required network resources for iot. In *Control, Electronics, Renewable Energy and Communications (ICCEREC), 2016 International Conference on*, pages 1–6. IEEE, 2016.

APPENDIX

A. ADDITIONAL MATERIALS

The experiment discussed in Chapter 4 is fully reproducible. All code used to perform it together with the latest version of the paper itself are available at https://gitlab.com/v45k0/iot-data-protocols.

Practical Assessment of Secure Multiparty Computation Frameworks

Jakub Wójcik
Advisor: Marcel von Maltitz
Seminar Future Internet WS2017/2018
Chair of Network Architectures and Services
Departments of Informatics, Technical University of Munich
Email: jakub.wojcik@tum.de

ABSTRACT

This paper describes, assesses and compares the secure multiparty computation frameworks FRESCO and Bristol SPDZ in terms of infrastructural differences, practicality and performance. It provides a step by step guide to performing computations with the frameworks. The measurement were performed in a virtual environment.

Bristol SPDZ is faster and uses less memory. It provides an easy way to define computations by writing simple Python scripts. FRESCO is more suited for rapid development by integrating into a Java development process. Its modular design makes it equipped for the future.

1. INTRODUCTION

Secure multi-party Computation is a cryptographic method to perform joint calculations of arithmetical functions by multiple parties without them getting to know each other's input values. Multiple names and abbreviations have been used, such as secure computation (SC) or multi-party computation (MPC), but in the paper the term used will be secure multi-party computation (SMPC).

What makes SMPC different than other forms of cryptography is the fact that it treats the participating parties as adversaries. The mathematical groundwork has been laid in the 90s, but SMPC was just considered theoretically for a long time. An important step was Shamir's secret sharing[16], which is a method to share a secret between multiple parties, so that together they can reconstruct it, while on their own they have only useless information. Equally important were proofs that secure protocols exist, such as in the paper by Ben-Or, Goldwasser and Wigderson[11], which is based on Shamir's secret sharing and is still in use nowadays. Only in recent years, due to better computation power and advancements in SMPC protocols, such as the development of the SPDZ protocol[13], practical implementations have become feasible.

A few actively developed and maintained frameworks have been created. Among them are e.g. Sharemind, FRESCO and Bristol SPDZ. Sharemind[6] is a company selling solutions based on Shamir's secret sharing to businesses wanting to compare themselves to others without releasing their private data. Because this is a typical application of SMPC, this paper will also use statistical functions to assess the frameworks.

This paper looks at two of the frameworks in detail: FRESCO and Bristol SPDZ. In section 2 each framework is described on it's own, while in section 3 they are compared to each other in respect to infrastructural differences, practicality and performance. The frameworks are considered production ready, if it is possible to perform thousands of computations in a negligible timespan, while also hiding the implementation details from the end user. This requirement makes the frameworks ready to use in stock exchanges or social media applications where results have to be provided instantly. Finally section 4 lists some ideas of future work to be done and section 5 offers a conclusion to the comparison of the frameworks.

2. CONSIDERED SMPC FRAMEWORKS

The considered frameworks FRESCO and Bristol SPDZ are both active open-source SMPC solutions. They are currently being developed on GitHub with the latest contributions on both being from November 2017.

The points listed above are all the similarities off the frameworks, virtually everything else about them is different. The frameworks use different tool-sets, follow different paradigms, and have been developed for different reasons.

2.1 FRESCO

The name FRESCO is an abbreviation of "A FRamework for Efficient Secure COmputation"[1]. The framework is an abstraction from the different specific SMPC protocols, that are called protocol suites in the context of FRESCO. Instead it is meant as a foundation which allows for any protocol suite to be used to compute the same functions.

FRESCO implements multiple protocol suites, such as BenOr-Goldwasser-Wigderson (BGW) that is secure for an honest majority. That means that if the majority of the parties doesn't deviate from the protocol, it is guaranteed that no one will obtain any additional information[11], only the computed function value.

Other suites are also implemented, e.g. SPDZ, but in this paper only the BGW suite will be taken into account because SPDZ is currently under active development[2]. For more information see section 4.

FRESCO is developed in Java 8. It is packaged with all required dependencies by Maven in a JAR which can be down-

doi: 10.2313/NET-2018-03-1_03

loaded from its website[5]. Computations are also meant to be written in Java by extending the `Application` class and implementing the sequence of computation steps to be performed. It can be embedded anywhere in a Java program, but the documentation provides a sensible default application that can be used to start FRESCO from the command line. Providing necessary configuration from the command line arguments, the application is run in one line.

The developer using FRESCO is able to build their own Java programs that call into FRESCO when SMPC is to be performed. They can bundle FRESCO with their program and deploy it as a single Java executable.

2.2 Bristol SPDZ

The SPDZ (pronounced "speedz"[10]) protocol has been introduced in 2011 and it uses different mathematical concepts than BGW, namely somewhat homomorphic encryption[13][7]. In has been improved twice. The first time in 2012 solving open problems of the original paper[12]. The second time in 2016 when MASCOT, a new protocol for the preprocessing phase, was introduced[15].

The special thing about SPDZ is that it is divided into two phases. The first one is called the offline phase, or rather the preprocessing phase as a connection between the parties is required. This phase generates triples, which are used to perform multiplications, and Message Authentication Codes (MACs), which use asymmetric cryptography to ensure that the values provided by the parties are correct. Both triples and MACs are used later during the online phase. In the online phase the actual computation is performed.

The online phase is designed to be fast, hence the name of the protocol. The computationally heavy stuff is performed during the preprocessing phase which can take place e.g. at night or on weekends, when no computations need to be performed. During the online phase MACs are needed for every step of the computation, while for every multiplication three previously generated triples are consumed. A lot of both has to be prepared beforehand.[15]

Bristol SPDZ is developed in C++ and Python 2. It has to be compiled from source and it depends on a few other libraries that have to be installed manually.[4]

The implementation of the SPDZ protocol is first and foremost a showcasing of the improved preprocessing phase MASCOT which has been proposed by the same people that develop the implementation.[7] As such it does not hide its internals and instead exposes every step as a separate executable or script that can be run.

3. ASSESSMENT OF THE FRAMEWORKS

The frameworks are compared and assessed in respect to practicality and performance.

As they have been developed with different use cases in mind, often no direct comparison is possible. Instead the execution of the chosen solutions is assessed.

3.1 Setup and methodology

In order to assess all relevant differences between the frameworks in a realistic setting, a fresh OS was set up in a virtual environment and all steps needed to achieve the goal of computing some statistical functions were counted towards practicality.

Used statistical functions. As performing statistics with SCMP is a common use case, in this paper statistical functions were used during the practicality and performance tests. The functions were chosen to include all of the basic arithmetical operations as it is possible to compute them using SMPC: addition, subtraction, multiplication and division.

The first one is the average function:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

The second one is variance:

$$\mathrm{Var}(X) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$$

Having the variance, it is easy to calculate the standard deviation by taking the square root. It is not yet possible to compute the square root with SMPC.

$$\sigma = \sqrt{\mathrm{Var}(X)}$$

Both frameworks were setup to calculate the functions in the same scenario. There were four parties configured, each of them having a single integer x_i as its secret input value. This section describes the effort it takes to configure the setup and measures the performance of the computations.

Virtual environment. The frameworks were tested in a virtual environment. The virtual environment was created using Vagrant and VirtualBox.

The chosen OS was `ubuntu/trusty64`[8] with 1024 MB of RAM. The VMs were running on a relatively modern processor with eight virtual cores and a clock speed of 1.2 GHz.

The performance of the framework was measured on the VMs with the Linux `time` command. The memory consumption was measured with `valgrind`. Although the values might not represent real conditions, they certainly are adequate to use them in comparisons with each other.

3.2 Key (infrastructural) differences

This section looks at the frameworks from an infrastructural point of view. Considered are the installation process, the dependencies, and the way the computations of the arithmetic functions have to be defined and performed. It also looks into the paradigm behind the frameworks.

The section also provides a step by step guide to performing computations with the frameworks.

doi: 10.2313/NET-2018-03-1_03

Installation and dependencies. Installing FRESCO is very straightforward. An prepackaged jar with all required dependencies can be downloaded from the FRESCO website[5].

Alternatively the framework can be compiled from source using Maven, which is only mildly more complicated. The only necessary dependencies system wise are the Java Development Kit version 1.8 and Maven in order to compile from source. Both of these are available out of the box on modern Linux distributions.

To be able to install Bristol SPDZ, first the C++ compiler `g++`, Python 2 and m4 have to be installed. They are available in the repositories of most modern Linux distributions. Additionally the Multiple Precision Integers and Rationals Library (MPIR) has to be installed. The instructions are available in the manual[14]. Simply run `configure` with the `-enable-cxx` flag to enable C++ compatibility.

To compile and install Bristol SPDZ the source code has to be downloaded from the repository[4]. The git submodule `SimpleOT`, which is necessary for MASCOT to work, has to be downloaded, too. In the `CONFIG` file, the flag `USE_GF2N_LONG`[1] has to be set to 1. Afterwards compilation is as simple as running `make`.

Defining computations. In order to define computations in FRESCO the `Application` class has to be extended and the `prepareApplication` method overridden. In the method body all of the steps required to compute the arithmetic function have to be defined using `ProtocolProducer`s. These steps form the protocol that will be performed with the specified protocol suite when running the application. Inserting the values into the computation and getting the output of the function has to be dealt with. There is a difference between secret and open values represented by types prefixed with an S or O. For example `SInt` and `OInt` are secret and open integer values, respectively. The secret numbers are effectively shared secrets. No party has enough information to know the values. Only together they can reconstruct it to create an open integer.

The order in which the steps will be computed has to be defined as well. Which operations will be performed sequentially and which ones in parallel. The developer of the are responsible for the optimization of the computation. It is possible to define one application as an extension of another.

Additionally the application must be run by calling the method `SCE.runApplication`. This can happen anywhere in the code. To compile the Java program the compiler has to know where it can find the packages of FRESCO. This is typically done by setting the `classpath` argument:
`javac -cp .:fresco-0.2-jar-with-dependencies.jar`

To define computations in Bristol SPDZ, a file with the extension `.mpc` inside of the subdirectory `Programs/Source/` has to be created. Inside the file the computation is defined by writing a Python script that operates on values of custom data types.

As with FRESCO in Bristol SPDZ there is a difference between secret and open data types. For example there are `sint` and `oint` representing secret and open integers, respectively. Inserting the private values into the computation and revealing the result has to be dealt with, but other than that the standard arithmetical operators can be used.

The definition of every computation has to be compiled by calling `compile.py` from the SPDZ directory with the filename of the program, but without the subdirectory. For the program to work with MASCOT, the flags `-p 128 -g 128` have to be passed. The flags set the sizes of the internally used data types. They are necessary because the compiler and MASCOT use different sizes by default. The compilation will generate multiple files inside of `Programs/`.

Paradigm. FRESCO clearly follows the paradigm of allowing to define computations independently of the protocol suite that will be used to perform the actual computation. Being developed purely in Java, it also can be neatly tied into existing Java applications. It is designed to be used by developers as a Java library to perform SMPC[3].

The paradigm behind Bristol SPDZ seems to be to create a working implementation of the newest improvement to the SPDZ protocol. It is meant as an showcasing of the MASCOT protocol. As such it does not focus on usability and instead gives the user full control over each internal step.

Performing computations. In order to perform a computation with FRESCO, the location of FRESCO has to be specified to the JVM in the `classpath` argument, again. When using the provided default program, which reads the parameters from the command line, additionally the protocol suite and the addresses of all the participating parties have to be specified. The number of the current party has to be specified, as well. Then all of the parties run the program having access only to their own private inputs. Party number one of two would run the following to use the BGW protocol suite:

```
java -cp .:fresco-0.2-jar-with-dependencies.jar App
-s bgw -Dbgw.threshold=1
-p 1:192.168.33.10:9001 -p 2:192.168.33.11:9002
-i 1
```

The program automatically connects to the other parties and performs the computation.

In order to perform an computation with Bristol SPDZ, multiple steps have to executed:

- Prepare the data for the online mode by running `Scripts/setup-online.sh 4 128 128`.
  The 4 in this context is the number of participating parties, while the 128 are the sizes of the used data types. They have to match the values provided to the compiler.
- Distribute the compiled program and the prepared data among the participating parties.

[1] Enables the use of a 128 bit GF($2n$) field that is needed by MASCOT.

doi: 10.2313/NET-2018-03-1_03

- Prepare a file `HOSTS` with the addresses of the parties. It is needed for MASCOT to work.
- Perform the MASCOT preprocessing phase by every party running `ot-offline.x -N 4 -p 0`. The flag `-N` defines the number of participating parties, while the flag `-p` defines which party one is. With the flag `-n` the number of generated triples can be set.
- Prepare the private input values in a separate file by running `gen_input_f2n.x` or `gen_input_fp.x` on each party. These tools encode the provided input values into a binary format.
- Start a server which will coordinate the communication between the parties by running `Server.x`. The server is only needed to establish a connection between the parties, the communication itself is peer to peer. The players have to be reachable by hostname.
- Run the online phase on every party.
  `Player-Online.x -lg2 128 -pn <server port> -h <server host> <player id>`.

3.3 Practicality

Practicality in this context means how well the framework can be used. This applies to its paradigm, how easy it is to write computations and how well the framework is documented.

3.3.1 FRESCO

FRESCO is designed for rapid development and it mostly fulfills this promise. One major drawback is the lack of up to date documentation.

The framework is undergoing some structural changes, so the documentation does not always line up with the actual implementation. This might change in the near future (cf. section 4). In the meantime it is probably better to learn using the framework by looking at working examples. The developers include a few demos with FRESCO.[1]

Defining computations. From an infrastructural point of view, defining computations in FRESCO works very well. Because of its nature as a Java library, defining new computations can be tied into an existing development process. It is possible to use FRESCO with IDEs such as Eclipse.

On the other hand, FRESCO forces the developer to define all the steps to perform the computation. They have to decide which steps have to be performed sequentially and which ones in parallel. This requires a deeper understanding of underling mechanisms and can get quite verbose. For example in the applications used in this paper the definition of the average computation has 15 lines, while the definition of variance has 30 line of code. This is just to define the steps of the computation. Any other code was not counted.

Additionally any optimization has to be performed manually. A handy feature showcased in the demos is that already defined computations can be reused to build more complicated ones by inheriting from another `Application` class.

Performing computations. This is where FRESCO shines.

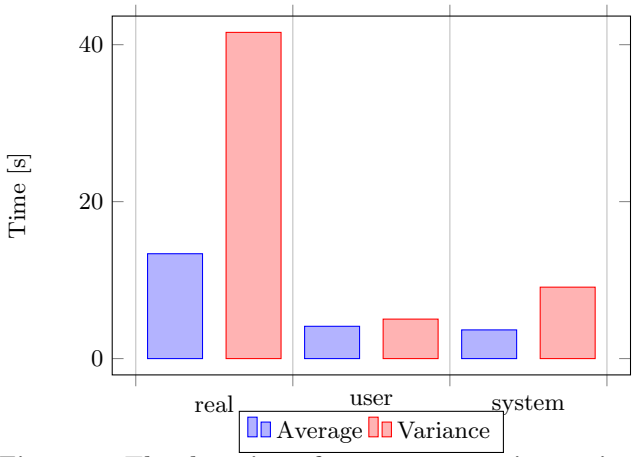


Figure 1: The duration of 1000 computations using FRESCO BGW.

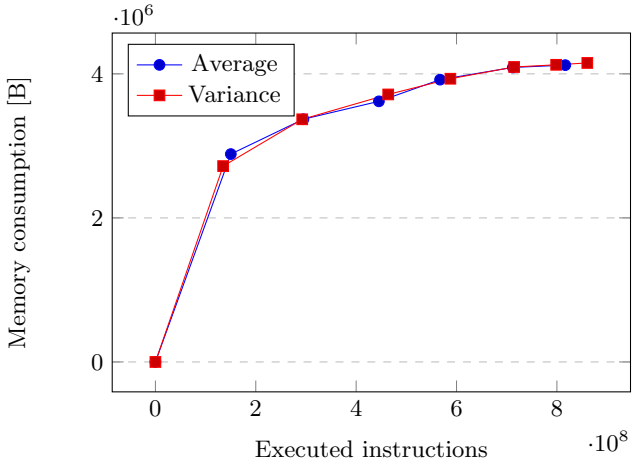


Figure 2: Memory consumption of one computation using FRESCO BGW.

Performing a computation is very straight forward. There is one executable, which takes all its configuration from command line parameters in one place. Only the protocol suite to use has to be chosen, but this offers a lot of flexibility. In the future, it should be even possible to use new protocols that are not implemented yet.[3]

3.3.2 Bristol SPDZ

Bristol SPDZ has been developed to showcase MASCOT. It has great features, but at the same time it is quite cumbersome to use.

Defining computations. This is where Bristol SPDZ shines.

Defining computations is very easy because it is just writing a Python script. The sequence in which the operations are computed and the optimization are inferred automatically.

The only drawback is a lack of a documentation about the custom data types. One can either look into the source code to learn which ones to use, or look at working examples or demos which are included in Bristol SPDZ.[4]

doi: 10.2313/NET-2018-03-1_03

Performing computations. Performing computations with Bristol SPDZ is very cumbersome. Every step and phase has to be run manually and a lot of files are generated in the process. Of course a the process can be simplified by using a shell script, when the desired configuration has been identified. In the repository there is even the script `Scripts/run-common.sh`[4] that can be used for this purpose, but it is not compatible with every setup.

The most annoying thing is that the same options have different names for the tools on different steps. The commands of each step also have their own unique formats. For example MASCOT uses a hosts file to communicate with the other parties while the online phase uses a server.

In order to be reachable by the hostnames, the file `/etc/hosts` has to be edited on every party. There should be a better way of getting a connection.

3.4 Performance

The performance of the frameworks was measured using the Linux `time` command. It returns three values: real, user and system. The value real stands for the actual duration the program was running. The other values show how much the time the processor spent in user and system mode, respectively.

It can be said that in general in user mode the arithmetical operations are performed. In system mode on the other hand it's mostly read and write operations and network communication. The sum of the times spent in both modes is not necessarily equal to the real time because the processor might spend some time on other calculations, such as running different processes. This is especially important considering that the tests were run in a virtual machine.

To get a better picture of the time the computation actually takes, and not to measure the startup time of the program, each computation is performed 1000 times.

In order to measure the amount of memory the programs consumed, the profiler `valgrind massif`[9] was used:
`valgrind -tool=massif -heap=yes -stacks=yes`

This command returns the size of the allocated heap and stack space during the runtime of the program. In the graphs the x-axis shows the number of instructions executed by the processor. This offers a overview over the memory consumption in different stages of the program, although the executed instructions are not necessarily proportional to the runtime of the process.

3.4.1 FRESCO

The performance of FRESCO is quite stable. As can be seen in figure 1 the time it takes to perform a 1000 simple computations is over 10 seconds. One thousand computations of average took 13.36 s, while the computations of variance took 41.57 s. The slightly more complicated computation of variance which has twice as many lines as average took nearly three times as long.

This does not feel like much when performed on its own, but it certainly means that performing a lot of computations in batch could take hours. It also means that FRESCO is not suitable for application in social media or stock exchanges where rapid response times are very important.

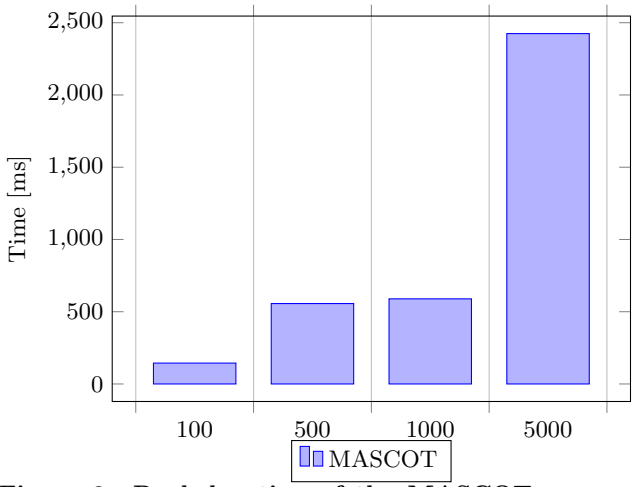


Figure 3: Real duration of the MASCOT preprocessing phase generating n triples.

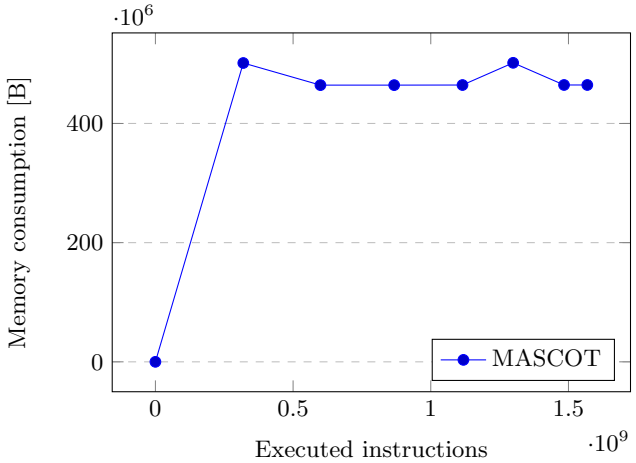


Figure 4: Memory consumption of MASCOT generating 5000 triples.

An interesting thing to note is that the computation time reported by FRESCO by enclosing the call to the `runApplication` method and the running time measured by the system are consistently about one second off. This probably means that one second is the startup time of the program and the Java Virtual Machine.

Memory consumption of FRESCO is reasonable. It constantly increased during the computations and reached about 4.0 MiB at the end. The progression can be seen in figure 2.

3.4.2 Bristol SPDZ

In Bristol SPDZ the performance measurement are divided between the preprocessing and online phase because they can be performed independently. The first one might be performed at night and the latter during the day when SMPC in needed.

doi: 10.2313/NET-2018-03-1_03

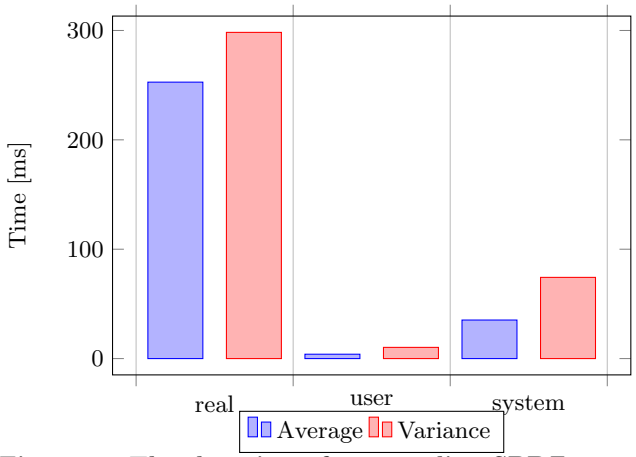


Figure 5: The duration of 1000 online SPDZ computations.

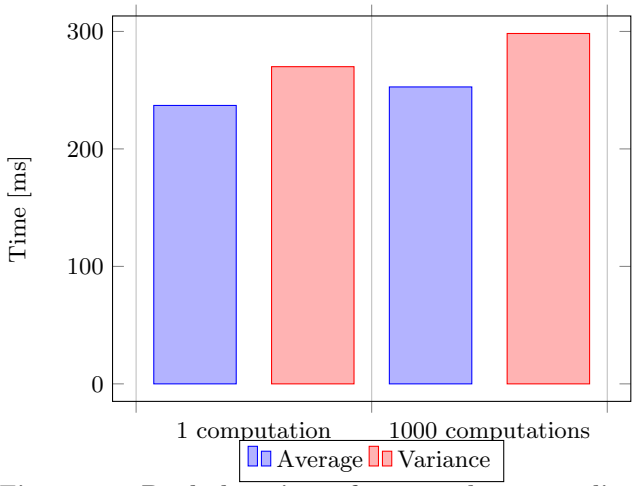


Figure 6: Real duration of one and 1000 online SPDZ computations.

Preprocessing phase. To measure the performance of MASCOT a different number of triples were generated. With more triples the running time rises significantly (see figure 3). It took 2.42 s to generate 5000 triples.

The memory consumption of MASCOT is very stable. While generating 5000 triple the memory usage was about 470 MB which is nearly half of the available RAM on the virtual machine (see figure 4).

Online phase. Compared to MASCOT, the online phase is very fast. One thousand computations of average took 253 ms, while the computations of variance took 298 ms (see figure 5).

Such speeds make it certainly possible to be used in application where speed is important, such as social media or stock exchanges. One has to keep in mind that this speeds are only possible if there are previously generated triples and MACs. When the parties run out of those, further computations become impossible. The generation of additional tripes is much slower (cf. figure 3).

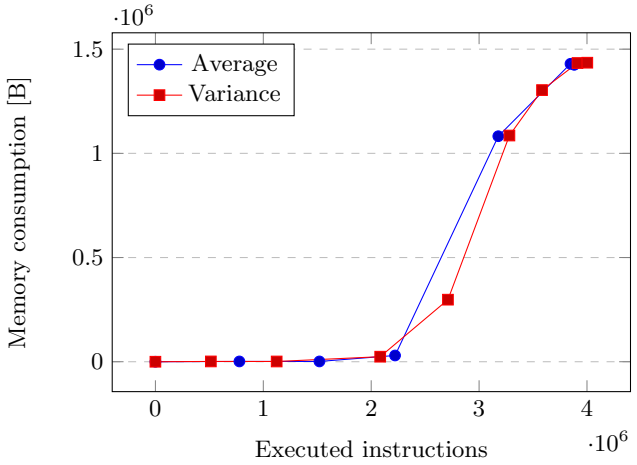


Figure 7: Memory consumption of one online SPDZ computation.

An interesting thing to note is that the duration time of one computation and 1000 computations are nearly identical (see figure 6).

The memory consumption of never exceeds 1.3 MiB (see figure 7). It is nearly zero for a long time, which according to the massif documentation[9] is a normal behavior of short running programs because most of the time they spend loading external libraries. This also explains the little difference between one and 1000 computations.

4. FUTURE WORK

Both of the frameworks are in active development with the latest contributions made in November 2017. This means that the information in this paper will not stay up to date for a long time. It is necessary to follow the development closely.

FRESCO. The current version of the framework FRESCO, version 0.2, is currently undergoing some structural changes. Because of this a lot of the features and protocol suites are not correctly documented.

Additionally a SPDZ protocol suite with the MASCOT preprocessing phase[2] that is used in Bristol SPDZ is currently being implemented by the FRESCO team. As it is not yet ready to be used, the SPDZ protocol suite has not been taken into account in this paper. Having two completely different implementations of the same protocol, it would certainly be interesting to measure the performance differences of both. Once the implementation is done, it will surely be worth to compare it to Bristol SPDZ and assess other differences, such as ease of use.

Actual network. All of the measurements in this paper were performed in a virtual environment. In order to understand how the frameworks perform in real circumstances, it would be a good additional analysis to perform the computations on an actual network.

doi: 10.2313/NET-2018-03-1_03

Other parameters that can be varied are the number of participating parties and the connection speed. It would also be interesting to see how the frameworks work over the Internet with the machines of the different parties being in different physical locations. This would probably be the closest scenario to a computation in real circumstances.

5. CONCLUSION

The frameworks are have been designed with different goals in mind. It is therefore impossible to deem one better then the other. Which one is better suited for a task depends heavily on which paradigm is better aligned with the goal of the task.

In terms of performance Bristol SPDZ clearly wins. It is faster and uses less memory. Even considering the sum of both phases it still is faster than FRESCO. Another place where Bristol SPDZ is strong, is defining the computations, which basically is the straightforward task of writing a Python script. Where it certainly lags behind is its lacks of a cohesive structure to the different execution steps.

FRESCO on the other hand wins when it comes to rapid development. It is easier to integrate into a development process and easier to run the implemented computations. Its modular design with different suites makes it better equipped for the future.

Application developed with both frameworks can be made production ready as defined in the introduction. The developer using FRESCO will have to make sure the number operations does not make their program slow, while the developer using Bristol SPDZ will have to put more effort into making SMPC work seamlessly in the background. Depending on the context both can be adequate compromises.

6. REFERENCES

[1] A framework for efficient secure computation. On GitHub `https://github.com/aicis/fresco`. Accessed: 2017-11-16.

[2] Implement proper spdz preprocessing. `https://github.com/aicis/fresco/issues/112`. Accessed: 2017-11-18.

[3] Introduction - fresco 0.2.0 documentation. `http://fresco.readthedocs.io/en/latest/intro.html`. Accessed: 2017-10-01.

[4] Multiparty computation with spdz online phase and mascot offline phase. On GitHub `https://github.com/bristolcrypto/SPDZ-2`. Accessed: 2017-11-16.

[5] Releases - fresco 0.2.0 documentation. `http://fresco.readthedocs.io/en/latest/releases.html`. Accessed: 2017-09-05.

[6] Sharemind | privacy enhancing technology for data-driven business. `https://sharemind.cyber.ee/`. Accessed: 2017-10-01.

[7] Spdz software | bristol university | department of computer science. `https://www.cs.bris.ac.uk/Research/CryptographySecurity/SPDZ/`. Accessed: 2017-11-18.

[8] Vagrant box ubuntu/trusty64. `https://app.vagrantup.com/ubuntu/boxes/trusty64`. Accessed: 2017-09-04.

[9] Valgrind user manual - massif: a heap profiler. `http://valgrind.org/docs/manual/ms-manual.html`. Accessed: 2017-09-30.

[10] What is spdz? part 1: Mpc circuit evaluation overview. `https://bristolcrypto.blogspot.de/2016/10/what-is-spdz-part-1-mpc-circuit.html`. Accessed: 2017-11-18.

[11] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault Tolerant Distributed Computation. *Proceedings of the 20th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 1–10, 1988.

[12] I. Damgard, M. Keller, E. Larraia, V. Pastro, P. Scholl, and N. P. Smart. Practical covertly secure mpc for dishonest majority – or: Breaking the spdz limits. Cryptology ePrint Archive, Report 2012/642, 2012. `http://eprint.iacr.org/2012/642`.

[13] I. Damgard, V. Pastro, N. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. Cryptology ePrint Archive, Report 2011/535, 2011. `http://eprint.iacr.org/2011/535`.

[14] T. Granlund and W. Hart. The multiple precision integers and rationals library. `http://mpir.org/mpir-2.7.2.pdf`, 19 November 2015. Accessed: 2017-09-30.

[15] M. Keller, E. Orsini, and P. Scholl. Mascot: Faster malicious arithmetic secure computation with oblivious transfer. Cryptology ePrint Archive, Report 2016/505, 2016. `http://eprint.iacr.org/2016/505`.

[16] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, Nov. 1979.

Ethics, Products, Top Lists - and their Use at Internet Measurement Conferences

Luca Ciprian
Advisor: Quirin Scheitle
Seminar Innovative Internet Technologies and Mobile Communications SS2017
Chair of Network Architectures and Services
Departments of Informatics, Technical University of Munich
Email: luca.ciprian@tum.de

ABSTRACT
In this paper, we evaluate ethical considerations, reproduction considerations, the use of geolocation products, and the use of top lists in the three major Internet Measurement Conferences. We take all the publication of the past three years into account, making a total of 260 evaluated papers. We further dive in deeper into subtopics for each of the four categories mentioned above, comparing differences and commonalities of papers and outline special characteristics of certain papers. In the end we can see that top lists are more frequently used than the other inspected subtopics. Additionally it stands out that the consideration of ethics and reproduction increases every year.

Keywords
Ethics, Reproduction, Geolocation, Top Lists

1. INTRODUCTION
Internet Measurement is conducted in order to understand and assess many different sectors of the web. Whether it is the security and privacy of users, the behavior of applications or simply analyzing the traffic, there are many fields that need thorough exploration to keep the information up to date. This paper examines the considerations of certain subtopics in Internet Measurement. Beginning with looking at Ethics, the focus is on the role ethical considerations play during Internet Measurement. In addition the opportunities to reproduce researchers' results are analyzed. Lastly the use of geolocation products and top lists during Internet Measurement research is evaluated. The base for this paper are the publications of annual Internet Measurement Conferences.

1.1 Conferences
Internet Measurement Conferences are held yearly and give researchers from all over the world the possibility to submit papers documenting their work. The best papers, selected by the conference's jury, are then presented by their authors at the conference. This paper focuses on three Internet Measurement Conferences: The "Internet Measurement Conference" (IMC), the "Passive and Active Measurement Conference" (PAM) and the "Network Traffic Measurement and Analysis Conference" (TMA). All the papers published at these three conferences in the course of the past three years are covered, totaling 260 papers.

2. ETHICS
Ethical considerations play an important role in internet measurement. Many topics call for an assessment previous to getting started. If you want to obtain certain data and measure things important to your research, you often affect servers and resources that belong to parties not being part of the research team and not knowing about the measurement.
It is common to ask for internal approval by the university's Institutional Review Board or Ethics Committee in order to conduct such a research. In that process you explain what you are intending to do, what the effects of your actions could be and what you will do to cover these effects.
This section will evaluate how many papers at the three Internet Measurement Conferences we focused on, consider the ethical points of their research.
The selected keywords for evaluating how many papers considered the ethics of their work in the past three years were: "ethic", "ethics", "ethical", "moral", "morally" and "righteous". If they appeared in a context that was not relevant for this paper, those sentences were ignored.

2.1 Ethics in scientific papers
Out of the 260 conference papers, 44 take into account the ethical consideration of their work. They clarify why their research was ethically acceptable and explain the measures taken to assure that. Mostly the measures included securing the privacy of people while collecting data. Additionally, some papers emphasize that it is important to give thought to ethics, especially being aware that potential harm could be caused to non-participants through research.
While reading the papers, it occurs that there are two ways the authors express their consideration of ethics. Nine of them briefly talk about how they try to sustain ethical correctness. For example: "Given the ethical considerations involved, we recruited only volunteers and took an informed consent from them all after explaining and demonstrating the entire process." [14] On the other hand 35 papers enlarge into ethics in an own chapter. They talk about the importance of ethical consideration in research in a more extensive manner and project it onto their work by explaining all the steps taken they felt were necessary. Independent of how comprehensive ethics were discussed, 17 papers additionally mention going through the process of obtaining their university's approval through an Institutional Review Board.

doi: 10.2313/NET-2018-03-1_04

Year	Total papers at all conferences	Papers	Ethical consideration	Chapter on Ethics	IRB approval
2015	87	4 (4.6%)	1	3	1
2016	92	16 (17.4%)	3	13	6
2017	81	24 (30.0%)	5	19	10

Table 1: Ethical Consideration in papers

2.1.1 References to publications on ethics

Some papers refer to other researcher's work regarding ethical measures in research: "As with any active scanning research, there are many ethical considerations at play. We followed the best practices defined by Durumeric et al. [...] and refer to their work for more detailed discussion of the ethics of active scanning." [32] Another example: "Partridge and Allman [16] propose to evaluate whether the active measurements themselves or the release of the resulting data can harm an individual." [11] Partridge and Allman [25] and Durumeric et al. [10] were named by most papers referring to other researcher's work.

2.1.2 E-Mails, websites and blacklists

In order to increase the transparency of their measurement, some research teams took additional measures. This paper explains how transparency was facilitated: "To minimize the intrusiveness of our active network measurements we implemented several procedures: [...] Second, we set up a website on the scanning machines which explains our measurement activity in detail. Third, we maintain a blacklist of hosts and networks which will not be scanned in any of our measurements. Throughout the experiment, we received one e-mail out of curiosity and another one asking to be blacklisted. We complied with the blacklisting request and did not probe this network anymore." [12] Other research teams took similar measures.[35, 2, 28, 4, 29]

2.1.3 Disclosure of security weaknesses

An other ethical issue, especially in security research, is the disclosure after finding a vulnerability. On the one hand it is important to inform the community about discovered security flaws, on the other hand it could cause damage to a company's reputation. This research group informed the developers about a bug they found in their product and advised a way to correct it. They then gave them time to fix the bug before disclosing: "Finally, once we confirmed the vulnerability, we notified both Periscope and Meerkat about this vulnerability and our proposed countermeasure (directly via phone to their respective CEOs). We also promised any disclosures of the attack would be delayed for months to ensure they had sufficient time to implement and deploy a fix." [36]

2.2 Conclusion

Table 1 shows the development of ethical discussions in papers for the past three years. It should be remarked that not all paper's topics are of content that needs an ethical consideration, wherefore the percentage values can't be compared to a target percentage of 100%.
While the total number of yearly papers at the conferences barely changed, the amount of papers mentioning ethical aspects, strongly increased. The percentage of papers discussing ethics per conference underlines this improvement.
Equally to the raising percentage, there is an increase in the Institutional Review Board's engagement into the ethics of a research. This overall improvement makes it clear that the Internet Measurement community considers ethical valuation in research evermore important.

3. REPRODUCTION

Reproducibility is very important in scientific research. It allows fellow researchers to acquire a more extensive understanding of a topic and gives them the opportunity to build their academic project on other researcher's work. The provided data also allows a thorough review and help's verifying the integrity of the published results.
This section focuses on papers that are careful of making their work easily reproducible. It checks the validity of internet links and the availability of data and code.
The selected keywords to find information on the reproducibility of a paper's work in the past three years were: "reproducible", "reproducibility", "reproduction", "reproduce", "reproduced", "reproducing".

3.1 Reproducibility of results

Searching the 260 papers for reproducibility, brings to light that 19 papers mention reproducible results. 15 of these papers have their focus on topics allowing direct reproduction through code and data. Four papers don't offer data, but address reproducibility or offer theoretical reproduction.
Looking at the papers with topics allowing active reproduction, 14 of the 15 papers specify a direct Internet link to required code and data for reproducing their results. The paper not disclosing a link for reproductional purposes is consciously not sharing the used code. The research topic focuses on network vulnerabilities, making it irresponsible and dangerous to reveal the used implementation to the public without limitations: "For ethical reasons, we do not publish our code: it will be shared with fellow researchers and interested anti-abuse projects on a request basis." [16]
From the 14 papers providing their code and data, each paper offers a valid link. All links directly lead to a website without redirections, with 13 links leading to an online hosting servive, whereas one of the links leads to a personal university website.[7] Table 2 illustrates these results.
From the four papers not offering reproducible data, one describes the reproduction of previous experiments.[26] Furthermore two papers present methods based on publicly available data making their presented methods easily reproducible.[21, 3] The last of these four papers addresses, that there could be legal consequences making certain data available to the public, especially malware. But on the contrary also saying that "[...] the sharing and reuse of ex-

links for reproduction	
working	14
dead	0
redirecting	0
personal site	1

Table 2: Papers offering Reproduction

Year	Total papers at all conferences	Papers with instructions on reproduction	%
2015	87	4	4.6%
2016	92	5	5.4%
2017	81	10	12.3%

Table 3: Reproduction in papers

isting datasets aids reproducibility, an important scientific goal."[33]

3.2 Conclusion

The importance of reproducibility in scientific research is indisputable, therefore scientific results should be repeatable, replicable and reproducible. Nevertheless many publications in computer science still lack reproducible information, making it more desirable in the academic community for the future.
Favorably all the links supporting reproduction are valid. However, the utilization of a personal website for reproduction purposes is not useful, since there is a high probability that the link won't be working in a couple of years.
Table 3 shows the amount of papers facilitating reproduction in the past years. The amount increased every year, especially in 2017. Even though the possibility of reproducing a result is strongly dependent on the paper's topics, it seems that the awareness to support reproducibility in research raises.

4. GEOLOCATION PRODUCTS

The use of IP-based geolocation databases to determine the physical location of a server or router is a well known practice in computer science research. Those databases are very convenient to use, making it possible to refer measured data to specific cities, countries or areas.
The market offers many different products, some are fee-based, others are free. Most of them provide particular levels of location accuracy, generally differentiating between city-level and country-level accuracy.
This chapter looks at various public and commercial IP Intelligence products that offer location based information and their use in research. It puts in contrast the use of paid and free products and evaluates which geolocation accuracy levels are most commonly prefered.
The selected keywords to obtain information on the utilization of geolocation products in research in the past three years were: "geolocation", "geo-location", "geo location", "IP2Location", "MaxMind", "GeoIP", "GeoLite", "DRoP", "NetAcuity" and "Neustar".

4.1 Geolocation product use

Of the 260 examined papers, 33 mention the use of geolocation databases, which are offered by four different providers:
-IP2Location [17]: IP2Location offers a comprehensive variety of geolocation databases. The price of a database depends on the features and informations provided. The free version, IP2Location LITE [18], is limited in accuracy and number of records compared to the commercial version.
-MaxMind: MaxMind also offers commercial and free options. GeoIP [22] is the commercial and GeoLite [23] is the free database. They are both available with city-level and country-level accuracy.
-Digital Element - NetAcuity Edge [9]: The NetAcuity databases are all fee-based databases. They are available in versions with different extent. All of the researchers that were found working with NetAcuity were using NetAcuity Edge
-Neustar IP GeoPoint [24]: Neustar's geolocation database is called Neustar IP GeoPoint and is fee-based.
Reading through the papers, it stands out quite fast that MaxMind is the predominant provider. In 26 of 36 cases the researchers selected a MaxMind geolocation database.

4.1.1 Free vs Paid and City vs Country

Table 4 demonstrates a detailed overview about the accuracy levels used with each database. Comparing the use of MaxMind's free database against it's paid database, the free one is slightly favored, with GeoLite being used ten times and GeoIP nine times. Seven papers mention the utilization of MaxMind for location purposes of their measures, but do not specify the used database. Investigating the accuracy, 13 of the 24 MaxMind users utilize city-level accuracy, whereas the country-level accuracy satisfies five research groups. The other papers do not provide information on the accuracy level used and it is not possible to deduce it from the context. Regarding the four papers using IP2Location, three choose the paid database and one uses the IP2Location LITE database. Likewise do three use city-accuracy and one uses country-accuracy.
Table 5 depicts, that most researchers favor using a paid geolocation database instead of a free one and city-level accuracy rather than country-level accuracy.

4.1.2 Reflected use

Some of the papers warn of the sole use of geolocation databases and address their inaccuracies: "Unfortunately, prior work has established that IP geolocation databases are often rather inaccurate [...]. To fix inaccuracies we use two other sources to manually estimate location for 5,172 unique, routable IP addresses observed over the course of the experiment." [30]
Another example, an entire paper on geolocation databases: "Our main contributions in this paper are: (1) we show that the studied databases have many inconsistencies, especially at city-level." [13]

4.2 Conclusion

Even though there are plenty of options, a lot of research groups utilize MaxMind's services for IP-based localization. Their database is by far more frequently used than the other databases. No paper mentions a reason for their decision. One cause could be that many researchers believe it to be the most accurate IP-based localization service. An other

doi: 10.2313/NET-2018-03-1_04

IP2Location		MaxMind GeoIP		MaxMind GeoLite		No Info		NetAcuity		Neustar	
4		9		10		7		5		1	
City	Country	City	Country	City	Country	City	Country	City	Country	City	Country
3	1	6	1	6	3	1	1	2	2	1	0

Table 4: Accuracy level used with every geolocation database

city vs. country accuracy		paid vs. free services	
city	19	paid	17
country	8	free	12
no information	9	no information	7

Table 5: geolocation accuracy and payment

reason for MaxMind's predominance could be, that because of the frequent use, it is the best-known and benefits from that scenario. Since there are many papers using geolocation services but not informing about the service they select, it is harder to assess the numbers. Focusing on the papers, that say which database is utilized, the paid version and city accuracy is preferred. This decision probably results from the researchers' need of city-accuracy and the paid databases being more precise. Although even the paid ones might have high inaccuracy on city level. Table 6 shows that the use of geolocation products decreased. Since it depends strongly on the research topics presented at the conferences, the decrease has no significance of geolocation databases being used less in internet measurement research.

Two papers did not disclose information about the uses geolocation databases [19, 5] One paper [29] used both, MaxMind GeoLite City and IP2Location. Another paper uses 4 databases (all 5 from Table 4 except Neustar). For this reason the numbers in Table 4 total 36.

5. TOP LISTS

To conduct website measurements it is favorable to know which the most popular websites on the internet are. That way the measurement is done on relevant pages and results in a meaningful outcome. Different companies focus on ranking websites, creating a list of the most popular websites, called top lists. An arbitrary number of the top websites on these lists is often selected by researchers to be included into their studies. Most top lists are created by ranking websites by page views and people using the website in a certain time period. This section evaluates the most commonly used lists in internet measurement research. Furthermore the number of selected top sites is examined and compared against each other. Lastly the researchers' motivation to chose a particular list is looked at.

Year	Total papers at all conferences	Papers using geolocation products	%
2015	87	15	17.2%
2016	92	9	9.8%
2017	81	9	11.1%

Table 6: Use of geolocation products

The selected keywords to find the top lists used by researchers in the past three years were: "Alexa", "Umbrella", "Quantcast", "SimilarWeb", "toplist", "top list" and "top domain".

5.1 Top list use

From the 260 examined papers, 56 use a top list as base for their Internet measurement. Table 10 shows the distribution over the past three years. These 56 papers utilize the top lists of only three different companies. Most common are the Alexa to plists [1]. Alexa offers a free of charge Top 1 Million global list. Furthermore Alexa provides a top list for every country it has enough data for, as well as a top list for various categories. The other two lists used in the papers are the Umbrella 1 Million list [6] and the Quantcast Top Websites list [27]. The Umbrella 1 Million list is a free list provided by Cisco with a global ranking of websites. The Quantcast Top Websites list offers a top list for every country and is also free of charge.
Surprisingly, every single research group used a top list ranked by Alexa for their analysis. The Umbrella list and the Quantcast list are only combined with an Alexa list in a single case respectively [2, 8], meaning that 54 of 56 research groups chose an Alexa list and two groups decided to amplify an Alexa list. One group used the Alexa Top 1000 domains from the traditional top lists and extended it with the websites of the 500 world's biggest companies according to Forbes Magazine.[37]

5.1.1 Comparison of individual top lists

Among the Alexa lists, the global lists were used more frequently than the country lists and the category lists. For the global lists the domains of the Top 1 Million list were applied the most, with 27 papers using it. Second are the domains from the Top 500 list, being used by seven papers. Table 7 shows which top domains were used by how many research groups.
The top list of a country was used in eight papers and the top list of a category was utilized by six research groups. Table 8 shows the different category lists used, while the News & Media category was applied twice . One paper uses the Top 500 list of each category [20], it is excluded of Table 8.

5.1.2 Motivation and reflected use

From 56 papers using Alexa's top domains, two papers describe what the top lists are and briefly elucidate the functionality of the website's ranking. [31, 15] None of the 56 papers explain the reasons for choosing an Alexa list over other top lists. Reflecting about the use of an Alexa top list, there are two papers mentioning a negative perception: "While Alexa provides high-quality rankings for the most popular sites, our experience has shown it to be less reliable

doi: 10.2313/NET-2018-03-1_04

Global top lists	
Top 500	7
Top 1K	5
Top 10K	3
Top 100K	4
Top 1M	27
No information	2

Table 7: Alexa's TopX domain usage

Category lists	
Adult	Top500
E-Mail	Top10K
News & Media	Top500
Region	Top500

Table 8: Category lists used with the amount of top domains

for the long tail of the distribution." [8]Another example: "We explicitly avoid using the Alexa ranking since it includes services which are questionable for some categories." [34]

5.2 Conclusion

It becomes clear that Alexa's top lists are prevalent in the Internet Measurement community. Almost all the papers mention the utilization of Alexa's ranking without explaining what it is and does, like it was natural to use that particular ranking. This implicitness is probably also the point why barely any paper describes the reason for selecting Alexa's top lists over another one. The Alexa Top 1 Million list is by far the most frequently used. Since all the papers use Alexa, the Top 1 Million list is also the most frequently used in research. Using many domains enables researchers to measure more data that still is significant, since it is ranked high. Looking at Table 10 there is a frequent use of top lists at Internet Measurement Conferences. Although they are not necessary for every research topic, their use has increased. Nevertheless, to state that more researchers use top lists for their internet scans seems difficult, since the use depends a lot on the research topics. Therefore the increment could just be arbitrary and not meaningful.

Year	Total papers at all conferences	Papers using top list	%
2015	87	16	18.4%
2016	92	21	22.8%
2017	81	19	23.5%

Table 10: Use of top lists

6. CONCLUSION

On the whole every category is represented with a good amount of papers regarding the distribution of research subjects and their topics. Table 9 depicts the complete distribution of papers for all the categories and all the conferences in the past three years. The percentage values refer to the total of 260 papers. It should be noted again, that none of the four categories could possibly be discussed in every paper. The percentage values are just an information, they should not be compared to a target percentage of 100%. Top lists are most commonly used in Internet Measurement, while the opportunity for reproduction is mentioned the least. As said in the previous chapters, this is a matter of topics and can not be compared. Ethics and reproduction opportunities strongly increased at the conferences in the past three years. These two categories are important to mention if they fit the research topic. More and more research teams acknowledge that fact. On the other hand the use of geolocation products and and top lists had a more random variation of mentions in the three years, most likely due to arbitrary topic decisions.

Conference	Papers total	Ethics	Reproduction	Geolocation	Top lists
TMA 2015	16	1	1	2	2
TMA 2016	16	4	1	1	3
TMA 2017	19	8	4	2	3
PMA 2015	27	0	1	4	4
PMA 2016	30	1	1	4	4
PMA 2017	20	3	2	2	5
IMC 2015	44	3	2	9	10
IMC 2016	46	11	3	4	14
IMC 2017	42	13	4	5	11
Total	**260**	**44**	**19**	**33**	**56**
Rate	-	**16.9%**	**7.3%**	**12.7%**	**21.5%**

Table 9: Complete statistics on all papers

doi: 10.2313/NET-2018-03-1_04

7. REFERENCES

[1] Alexa Top 500 sites on the web. https://www.alexa.com/topsites.

[2] J. Amann, O. Gasser, Q. Scheitle, L. Brent, G. Carle, and R. Holz. Mission Accomplished? HTTPS Security after DigiNotar. IMC, nov 2017.

[3] S. Benitez-Baleato, N. B. Weidmann, P. Gigis, X. Dimitropoulos, E. Glatz, and B. Trammell. Transparent estimation of internet penetration from network observations. PAM, 2015.

[4] R. Beverly. Yarrp'Ing the Internet: Randomized High-Speed Active Topology Discovery. IMC, 2016.

[5] I. N. Bozkurt, A. Aguirre, B. Chandrasekaran, P. B. Godfrey, G. Laughlin, B. Maggs, and A. Singla. Why Is the Internet so Slow?! PAM, 2017.

[6] Cisco Umbrella. https://umbrella.cisco.com/blog/2016/12/14/cisco-umbrella-1-million/.

[7] G. Comarela, E. Terzi, and M. Crovella. Detecting Unusually-Routed ASes: Methods and Applications. IMC, 2016.

[8] J. DeBlasio, S. Savage, G. M. Voelker, and A. C. Snoeren. Tripwire: Inferring Internet Site Compromise. IMC, 2017.

[9] Digital Element - NetAcuity Edge Premium. https://www.digitalelement.com/solutions/netacuity-edge-premium/.

[10] Z. Durumeric, E. Wustrow, and J. A. Halderman. ZMap: Fast Internet-wide Scanning and Its Security Applications. USENIX Security 13.

[11] O. Gasser, F. Emmert, and G. Carle. Digging for Dark IPMI Devices: Advancing BMC Detection and Evaluating Operational Security. TMA 2017.

[12] O. Gasser, Q. Scheitle, S. Gebhard, and G. Carle. Scanning the IPv6 Internet: Towards a Comprehensive Hitlist. TMA 2016.

[13] M. Gharaibeh, A. Shah, B. Huffaker, H. Zhang, R. Ensafi, and C. Papadopoulos. A Look at Router Geolocation in Public and Commercial Databases. IMC, Nov 2017.

[14] P. Gupta, M. Patel, and K. Chebrolu. *Cutting Internet Access Costs Through HTTPS Caching: A Measurement Study*. PAM 2017.

[15] T. Halvorson, M. F. Der, I. Foster, S. Savage, L. K. Saul, and G. M. Voelker. From .academy to .zone: An Analysis of the New TLD Land Rush. IMC, 2015.

[16] L. Hendriks, R. de Oliveira Schmidt, R. van Rijswijk-Deij, and A. Pras. On the Potential of IPv6 Open Resolvers for DDoS Attacks. PAM, 2017.

[17] IP2Location. https://www.ip2location.com/.

[18] IP2Location LITE - Free Databases for Download. https://lite.ip2location.com/.

[19] M. Javed, C. Herley, M. Peinado, and V. Paxson. Measurement and Analysis of Traffic Exchange Services. IMC, 2015.

[20] A. J. Kaizer and M. Gupta. Characterizing Website Behaviors Across Logged-in and Not-logged-in Users. IMC, 2016.

[21] A. Marder and J. M. Smith. MAP-IT: Multipass Accurate Passive Inferences from Traceroute. IMC, 2016.

[22] MaxMind GeoIP2 Databases. https://www.maxmind.com/en/geoip2-databases.

[23] MaxMind GeoLite2 Free Downloadable Databases. https://dev.maxmind.com/geoip/geoip2/geolite2/.

[24] Neustar IP GeoPoint. https://www.risk.neustar/ip-intelligence/ip-address-data.

[25] C. Partridge and M. Allman. Ethical Considerations in Network Measurement Papers. *Commun. ACM*, Sept. 2016.

[26] D. A. Popescu and A. W. Moore. Reproducing Network Experiments in a Time-controlled Emulation Environment. TMA, 2016.

[27] Quantcast Top Websites. https://www.quantcast.com/top-sites/.

[28] Q. Scheitle, O. Gasser, M. Rouhi, and G. Carle. Large-Scale Classification of IPv6-IPv4 Siblings with Variable Clock Skew. TMA, 2017.

[29] Q. Scheitle, O. Gasser, P. Sattler, and G. Carle. HLOC: Hints-Based Geolocation Leveraging Multiple Measurement Frameworks. TMA, 2017.

[30] P. Schmitt, M. Vigil, and E. Belding. *A Study of MVNO Data Paths and Performance*. PAM. 2016.

[31] P. Snyder, L. Ansari, C. Taylor, and C. Kanich. Browser Feature Usage on the Modern Web. IMC, 2016.

[32] D. Springall, Z. Durumeric, and J. A. Halderman. Measuring the Security Harm of TLS Crypto Shortcuts. IMC, 2016.

[33] D. R. Thomas, S. Pastrana Portillo, A. Hutchings, R. N. Clayton, and A. R. Beresford. Ethical issues in research using datasets of illicit origin. 2017.

[34] S. Traverso, M. Trevisan, L. Giannantoni, M. Mellia, and H. Metwalley. Benchmark and Comparison of Tracker-blockers: Should You Trust Them? TMA, 2017.

[35] B. VanderSloot, J. Amann, M. Bernhard, Z. Durumeric, M. Bailey, and J. A. Halderman. Towards a Complete View of the Certificate Ecosystem. IMC, 2016.

[36] B. Wang, X. Zhang, G. Wang, H. Zheng, and B. Y. Zhao. Anatomy of a Personalized Livestreaming System. IMC, 2016.

[37] H. Zhang, M. Gharaibeh, S. Thanasoulas, and C. Papadopoulos. BotDigger: Detecting DGA Bots in a Single Network. TMA, 2016.

Text Mining on Mailing Lists: Tagging

Florian Haimerl
Advisor: Daniel Raumer, Heiko Niedermayer
Seminar Innovative Internet Technologies and Mobile Communications WS2017/2018
Chair of Network Architectures and Services
Departments of Informatics, Technical University of Munich
Email: flohai@mytum.de

ABSTRACT
Keywords are an important metric for categorizing or clustering documents. If the number of documents in a collection of documents exceeds a certain amount, it becomes unfeasible to manually assign these keywords. One of these is the collection of mails in the IETF mailing lists.
For collections of that size, the only feasible way to assign keywords to all documents, is to automatically extract them. In this paper, we deal with this challenge in two different approaches. We implement both approaches to automatically extract keywords from the mails in the IETF mailing lists and compare their performance.

Keywords
Tagging, keyword extraction, IETF, mailing lists, RAKE, TF-IDF

1. INTRODUCTION
In the effort to standardize the Internet, the Internet Engineering Task Force (IETF) organizes itself through mailing lists [1]. Consisting of over 2 million mails, the amount of data these mailing lists represent is huge. To be able to bring order into these mails and to further analyze them, finding keywords that best describe the contents of a mail would be helpful. Since the number of mails is too vast to do this manually, this paper analyses and compares approaches for automatic keyword extraction.
For this purpose, we start with analyzing the problem in section 2. We describe the IETF mailing lists and the database, we were working on. Then, in section 3, we describe different approaches for automatically extracting keywords from text. We focus on RAKE, a ready to use Python library [9] and on a more elaborate approach that utilizes the NLTK [6] to compute TF-IDF values [12]. Afterwards, we describe our implementation of these approaches in section 4. In section 5, we compare the implemented approaches and analyze how well they were suited for the task. We finish by mentioning related work in section 6 and we sum up this paper with a conclusion in section 7.

2. PROBLEM ANALYSIS
In this section, we we will analyze the status quo and give an overview of the data we are working on.

2.1 IETF Mailing Lists
The Internet Engineering task force is an organization, that is tasked with creating documents standards and best current practices for the Internet [1]. The IETF is mainly organized through open mailing lists, anyone can contribute to. Since there are over 1000 lists [16], both active and inactive that contain a total of more than 2.1 million mails, organizing them or extracting scientific data from them is a challenge.

Table 1: Frequency of Special Characters

column	type	constraint
file	text	
key	integer	not null
date	timestamp w timezone	
date_local	timestamp	
sender_addr	text	
receiver	text	
subject	text	
messageid	text	not null
inreply	text	
spam	boolean	default false
spamscore	numeric	
sender_name	text	
person	bigint	
fast_person	bigint	

2.2 The Database
To work with and analyze the data contained in the IETF mailing lists, the Chair of Network Architectures and Services at the Technical University of Munich [14] has a postgreSQL database that contains the mails and the mailing lists of the IETF. For this paper, the relevant part of that database is the *mails* table (table 1). It contains a *file* and a *key* column. With those, it is possible to uniquely identify the file that contains the mail. Furthermore, we only made use of the *spam* column, so we could ignore spam mails.

2.3 Goal
Our goal is to find keywords that best describe the contents of the mails. Therefore, we need to find a metric to assign a score to every word that occurs in a mail and then select

doi: 10.2313/NET-2018-03-1_05

the top scoring words.
We are not interested in key phrases, since single words appear to be more useful for further analyzing or processing the mails. This way, one can for example cluster the mails by these keywords.

3. APPROACHES

There are many different approaches to automatically extract useful keywords from text [2]. Since implementing and comparing all of them would go beyond the intended scope of this paper, we chose to go in depth for two promising approaches.

3.1 RAKE

RAKE [9] is a Python implementation of the Rapid Automatic Keyword Extraction algorithm, proposed by Rose, et al. [10]. The algorithm first identifies candidate keywords, before scoring those keywords based on their frequency in the document. This approach is completely corpus independent, which makes it efficient for a dynamic corpus like our database, that grows with every mail that is sent to one of the mailing lists.
Even though RAKE is also capable of identifying key phrases, as mentioned in section 2.3, we will focus on identifying single keywords. This is possible, since there are multiple parameters, that can be used to configure RAKE. This way, besides the maximum number of words in a key phrase, one can configure a stop word list, the minimum number of chars in a keyword and the minimum number of occurrences of the key phrase in in the document. Furthermore, there are some parameters that are only important for key phrases with more than one word.
We chose RAKE as one of our candidates because in contrast to our second approach, which we will explain in the next section, it is an out of the box approach, that is completely corpus independent.

3.2 TF-IDF with NLTK

Our second approach, is primarily based on the TF-IDF metric. Leskovec, et al. call it a "measure of word importance"[4]. It is the product of the document frequency tf and the inverse document frequency idf. Here, tf is the number of times, a word occurs in the document under inspection, whereas idf is the proportion between the number of documents in the corpus and how many of those documents contain said word. In short, idf measures the importance of a word in the corpus. An in depth explanation on the TF-IDF metric is provided by Leskovec, et al. [4].
With this approach, we are more flexible to make modifications to suit our goal. Thus, we can use the Natural Language Toolkit (NLTK) to improve the keywords, we receive with TF-IDF [13]. The NLTK "is a leading platform for building Python programs to work with human language data" [6]. This way, we can use a tokenizer and a lemmatizer, provided with the NLTK. The lemmatizer makes sure, words with the same stem can be treated as the same. For example, words like *write* and *writing* will be seen as equal and *writing* will be replaced by *write*. Also, using the NLTK offers the possibility to further narrow down the selected keywords to only use a certain type of words, like nouns, or to remove some type of words, like human names.
We chose this approach, since it is a good contrast to RAKE in some major points. While RAKE only works on a single document, the TF-IDF is computed on the whole document corpus. This way, the keywords promise to be more relevant in context of the corpus. Also, since this approach is less out of the box, it is possible to further refine it.

4. IMPLEMENTATION

We implemented the keyword extraction in Python, since the libraries we used were written in Python. Also, the scripts, the chair provided for working with the database, were also written in this language.
For us, the most important of these scripts are in the *mail* package. After getting the identifiers of the mails we want to analyze from the postgreSQL database, we get the actual text from the mail with the *mail.get_message* function. Furthermore, we can use *mail.strip_html* to remove HTML tags from HTML mails.
To store the keywords we extract from the mails, we introduce two new tables. The table *keyword* will contain each keyword the script finds, exactly once. Furthermore, the table *mails_to_keyword* (table 2) will contain relations between these keywords and the mails in the *mails* table. For each one of these relations, we also store the algorithm, that was used and the score the keyword reached. By storing the keywords this way, instead of storing a comma separated list in the *mails* table, processing the keywords will be easier. For example, clustering the mails by keywords can easily be done by grouping them with SQL.

After starting the Python command line tool, the user enters a short configuration dialog. Here, the user can make decisions like which approach (section 3) to use or if the script should be run in *debug* mode. In debug mode, the keywords will be printed to the command line instead of being written to the database.
For the actual keyword extraction, the implementation of the two approaches has some major differences, even though we made them fit to the same structure. First, an *init_analyzer* function initializes an analyzer object. Then a *handle_mail_cursor* function loops over the mails and extracts the keywords with the analyzer.
Next, we will describe how those functions work in the different approaches.

Table 2: mails_to_keyword

column	type
messageid	TEXT REFERENCES mails
keywordid	INTEGER REFERENCES keyword
score	INTEGER
algorithm	VARCHAR(6)

4.1 RAKE

Since RAKE is an out of the box solution, in this case the *init_analyzer* function is fairly straightforward.

```
return rake.Rake("Rake/SmartStoplist.txt",
                 3, 1, 2)
```

This creates a *Rake* object, that uses *Rake/SmartStoplist.txt* as a stopword list. As candidate words, it considers all words with at least three characters that occur at least twice in

doi: 10.2313/NET-2018-03-1_05

the document. The third argument can be used to define the number of words in a keyphrase. In our case, we set it to one, since we are only looking for keywords.

Thanks to the way RAKE works, *handle_mail_cursor* is similarly simple. We only have to call the *Rake* object's run method, handing over the text we want to extract the keywords from as the only argument.

```
tags = analyzer.run(text)
```

This returns a list of tuples, containing each word in the document and it's score. The higher this score is, the better the word is suited as a keyword.

4.2 TF-IDF with NLTK

This approach is more complicated. We use Python's machine learning library sklearn [11] to compute the TF-IDF values of the documents.
Therefore, in *init_analyzer* we have to train the analyzer on a dataset. We use a subset of the mails in the database (if possible, all mails) as the training set.

```
vectorizer=TfidfVectorizer(tokenizer=tokenize,
                           decode_error='ignore')
vectorizer.fit_transform(mails)
return vectorizer
```

We initialize the tfidfVectorizer with a custom tokenizer, so we can use the NLTKs stopword list and lemmatizer. Then, we call $vectorizer.fit\_transform()$ with our training set. This method computes *idf* values for the vocabulary in the training set.

Once the tfidfVectorizer is initialized, we can extract keywords from each mail:

```
tags = analyzer.transform([text])
feature_names = analyzer.get_feature_names()
for i in values.nonzero()[1]:
    return_dict[feature_names[i]]=values[0,i]
return return_dict
```

$analyzer.transform$ computes the TF-IDF values of all terms in the mail. $analyzer.get\_feature\_names$ returns a mapping from the terms' ids to the actual terms, since the matrix returned by $analyzer.transform$ only contains ids and not the actual terms. We then create a dictionary, that maps the TF-IDF values to the terms. Finally, the best suited keywords for each mail, are the ones with the highest TF-IDF values.

5. EVALUATION

For Evaluation, we consider two different metrics. Under *runtime*, we compare the time the approaches need for the different parts of the algorithm. Under *Results* we compare the keywords, the two algorithms extracted, by looking at false positives and false negatives.

5.1 Runtime

When analyzing the runtime of the two approaches, we have to differentiate between the *init_analyzer* and the *handle_mail_cursor* phases, we described in section 4. In this section, we will evaluate those two phases separately and then give a short combined verdict.

5.1.1 initAnalyzer

As we already described in section 4, the two approaches differ greatly in this part of the algorithm. While in case of RAKE the initialization phase only consists of setting a few configuration values, for TF-IDF we have to do a lot more. The initialization phase of TF-IDF is a training phase, where the analyzer has to analyze all mails in the training set. This factor has a huge effect on the runtimes of the initialization phase.
In table 3, we show the runtime for different sizes of training sets.

Table 3: runtime initAnalyzer in seconds

Algorithm	10 entries		100 entries		1000 entries	
	total	entry	total	entry	total	entry
RAKE	0.001	0	0.001	0	0.001	0
TF-IDF	2.834	0.283	8.250	0.082	33.096	0.033

For RAKE, the runtime of the initialization phase is negligible. It takes under one millisecond and does not depend on the size of the training set. This is exactly the result one would expect, since RAKE only does a few configurations in this phase.
Also as expected, TF-IDF is much slower in this phase, since it has to train its model and compute all the IDF values for the vocabulary of the training set. As one can see in table 3, the time the initialization phase takes for one entry decreases, the more mails we have in the training set. This shows, that even though the TF-IDF implementation takes a lot of time, especially compared to RAKE, it scales really well for a bigger training set.

5.1.2 handleMailCursor

In the main phase of the algorithm, the actual keyword extraction, we expect the timings to behave similarly in both algorithms. In table 4, we see that this is roughly the case.

Table 4: runtime handleMailCursor in seconds

Algorithm	10 entries		100 entries		1000 entries	
	total	entry	total	entry	total	entry
RAKE	0.065	0.007	0.876	0.009	6.194	0.006
TF-IDF	0.354	0.035	7.556	0.076	82.790	0.083

Both algorithms do not have big changes in the amount of time it takes to extract keywords from one mail. TF-IDF get's slightly slower, the more mails we process. We assume that this is due to the fact, that more mails lead to a bigger vocabulary. This seems to slightly slow down the process of calculating TF-IDF scores.
The general difference in performance between the two approaches is in a big part because of the tokenizer, that uses the NLTK. Here, lemmatizing the documents takes up a noticeable amount of time.

5.1.3 Verdict

It is pretty obvious, that RAKE is much faster than our TF-IDF implementation, due to it scoring words independently of other documents. But the TF-IDF approach also does not explode with a bigger data set. In this approach, we can win a lot of time, if we save the analyzer (for example with

doi: 10.2313/NET-2018-03-1_05

Python's *Pickle* module [8]) after it has been initialized, so we can reuse it in the next run.

5.2 Results

Both approaches are able to yield reasonably good results. In appendix A, we attached an arbitrarily selected example mail from the database. This mail is a more or less typical example for many of the mails we find in the database.

Table 5: 10 best scoring Keywords for appendix A

RAKE		TF-IDF	
sip	1.25	presence	0.476
refresh	1.0	sip	0.431
upload	1.0	upload	0.259
proposed	1.0	document	0.203
expiration	1.0	register	0.199
		method	0.161
		expiration	0.137
		generalization	0.137
		refresh	0.130
		use	0.127

As one can see in table 5, both approaches yield a useful selection of keywords. In the next sections, we will go a bit more into detail, by looking at false positives and false negatives the algorithms produce. We will also delve into a few further observations.

5.2.1 False Positives

In this case, RAKE yields really good results. The only keyword that subjectively seems like a false positive, is "proposed". On the other hand, TF-IDF produced a few keywords that seem like bad choices (i.e. "refresh" or "use") , but those do have low scores.

5.2.2 False Negatives

Here, RAKE fails to produce the top scoring keyword of the TF-IDF approach. This is most likely due to the fact, that RAKE only returns keywords that reach a certain threshold.

5.2.3 Further observations

We made a few observations with other documents, that are not visible in our example document. Since RAKE does not lemmatize the documents or use some other kind of natural language processing like the NLTK, we sometimes receive useless keywords like parts of URIs or email addresses.
Also, the lack of lemmatizing leads to RAKE sometimes returning the same keyword twice, with only slight differences like one being the plural of the other one.
One phenomenon we observe in both approaches is that often, human names are extracted as keywords. This is something that could be prevented in the TF-IDF approach, by optimizing the tokenizer.

5.3 Overall verdict

In general, we see RAKE yields a reasonably good ratio between false positives and false negatives and also outperforms the TF-IDF approach by far, when it comes to runtime. But, as we explained in section 5.2.3, RAKE also produces a lot of useless keywords. This is a big trade off, considering the fact that TF-IDF produces comparably good or even better results, especially when it comes to not missing any important keywords.
RAKE's big trade off could be fixed by modifying the RAKE implementation [9] and adding the option to use a custom tokenizer.

6. RELATED WORK

As we already mentioned in section 3, there are a lot of different approaches to extracting keywords from text. One famous approach is the textRank algorithm [5] by Rada Mihalcea and Paul Tarau. This is a graph based algorithm, that has similarities to Google's pageRank algorithm [7]. TextRank is also able to extract the most important sentences from a document, which can be used for automatic summary generation.
Another noteworthy approach is the KEA algorithm [15]. For this algorithm, a Java library is available. What makes this approach different to the ones we implemented, is that it requires a training set, that contains documents with manually assigned keywords.
Finally, we want to have a short look at another RAKE implementation by Sujit Pal [3]. He used the NLTK to streamline the standard RAKE implementation. For example, he used the NLTK tokenizer instead of RAKE tokenizing the text by itself.

7. CONCLUSION

In this paper, we did not introduce new approaches to keyword extraction. Instead, the aim was to find a good way of extracting keywords from the mails in the IETF mailing lists, by utilizing some of the existing approaches for keyword extraction. Therefore, we selected the two approaches we deemed most suitable for this task: The Rapid Automatic Keyword Extraction algorithm (RAKE) [10] and an approach, that computes TF-IDF scores [12] and optimizes this by employing the Natural Language Toolkit [6] for lemmatizing, tokenizing and stop word removal.
We described an implementation of these approaches and compared the results of both approaches based on those implementations. This comparison shows, that in the best case, RAKE would be the better choice, most of all due to its much shorter runtime. But, since in some cases RAKE produces unusable results, we reach the conclusion that in the current implementation, the TF-IDF approach is to be preferred.
In section 6, we mentioned an approach that modifies RAKE, by using the NLTK for tokenizing [3]. This approach can serve as an example for how to modify RAKE to better suit our needs.

8. REFERENCES

[1] H. Alvestrand. A mission statement for the ietf. BCP 95, RFC Editor, October 2004.

[2] Getting Started with Keyword Extraction. `http://textminingonline.com/getting-started-with-keyword-extraction`.

[3] Implementing the RAKE Algorithm with NLTK. `http://sujitpal.blogspot.de/2013/03/implementing-rake-algorithm-with-nltk.html`.

[4] J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of massive datasets*. Cambridge university press, 2014.

doi: 10.2313/NET-2018-03-1_05

[5] R. Mihalcea and P. Tarau. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004.

[6] Natural Language Toolkit. `http://www.nltk.org/`.

[7] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[8] pickle âĂŤ Python object serialization. `https://docs.python.org/3/library/pickle.html`.

[9] RAKE - A python implementation of the Rapid Automatic Keyword Extraction. `https://github.com/zelandiya/RAKE-tutorial`.

[10] S. Rose, D. Engel, N. Cramer, and W. Cowley. Automatic keyword extraction from individual documents. *Text Mining: Applications and Theory*, pages 1–20, 2010.

[11] scikit-learn: Machine Learning in Python. `http://scikit-learn.org/stable/`.

[12] Tf-idf: A Single-Page Tutorial. `http://www.tfidf.com/`.

[13] TF-IDF with scikit-learn. `http://www.bogotobogo.com/python/NLTK/tf_idf_with_scikit-learn_NLTK.php`.

[14] TUM Informatics - Chair of Network Architectures and Services. `https://www.net.in.tum.de/homepage/`.

[15] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning. Kea: Practical automatic keyphrase extraction. In *Proceedings of the fourth ACM conference on Digital libraries*, pages 254–255. ACM, 1999.

[16] www.ietf.org Mailing Lists. `https://www.ietf.org/mailman/listinfo/`.

doi: 10.2313/NET-2018-03-1_05

APPENDIX

A. EXAMPLE EMAIL

```
hello, my name is ****,
i sent this mail in the simple wg...

which is your opinion ?


> i think that this question: "upload presence document" is a
> critical issue.
>
> there isn't a standard procedure to upload and to modify the
> presence document, but it is the source of the presence status ...
> if the UA... if the PUA change its media capability, or
> change its status, it MUST refresh the presence document.
>
>
> the upload must be made through the protocol sip!!!
> and not push the presence doc via another protocol, such as HTTP POST.
>
>
> I agree with ******* when he says in the draft
> "Requirement for Pubblication of SIP related service data"
> : "... it is felt that the SIP REGISTER request is NOT the
> appropriate mechanisme for handing this loading od SIP
> service information..." but i'm not sure that a
> generalization of the REGISTER function, as proposed in the
> draft, is a good idea
>
>
> The use of the method REGISTER, for the upload,introduces one
> series of problems, just one example: the expiration of the
> Presence document and the expiration of the current
> communication addresses (i.e. Contact Addres) are different...
>
> I have instead a proposed other:
> the use of method NOTIFY...
> this method is already used for send the presence state to
> a particular subscriber...
> the its generalization for the upload or refresh of Presence
> Document is very simple and has the advantage of not
> modify/generalization a very important message for sip as REGISTER...
>
>

_____________________________________________
Sipping mailing list  http://www1.ietf.org/mailman/listinfo/sipping
This list is for NEW development of the application of SIP
Use sip-implementors@cs.columbia.edu for questions on current sip
Use sip@ietf.org for new developments of core SIP
```


doi: 10.2313/NET-2018-03-1_05

Kryptocoin Diebstähle

Jan Luca Pawlik
Betreuer: Marcel von Maltitz
WS2017/2018
Seminar Innovative Internet-Technologien und Mobilkommunikation
Lehrstuhl für Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: luca.pawlik@tum.de

KURZFASSUNG

Mit dem steigenden Wert und der steigenden Beliebtheit von Kryptowährungen steigen auch die Anreize für Hacker, Betrüger und andere Kriminelle, sich mit diesen zu beschäftigen. Um einen Überblick über die Gefahrenpotentiale der verschiedenen Diebstahlszenarien zu erhalten, ist eine Analyse von bisherigen Fällen bezüglich verschiedener Charakteristika interessant.
Dazu werden insgesamt 55 Diebstähle untersucht, die zwischen 2011 und 2017 stattfanden und zur Erstellung einer Taxonomie verwendet.

Schlüsselworte

Bitcoin, Ethereum, MtGox, ICO, Diebstahl, Betrug, Hack, Blockchain

1. EINLEITUNG

Die Vergangenheit hat gezeigt, es gibt eine Vielzahl von Angriffen auf die mittlerweile sehr heterogene Kryptocoinservice-Szene. Gerade wegen des aktuellen Hype um Bitcoins, Ether und andere Kryptowährungen und den teilweise überforderten Anbietern von Crpytocoinservices [21] bietet sich eine Untersuchung bereits erfolgter Diebstähle von Coins an.
Zweck dieses Papers ist es, einen Überblick über die verschiedenen Angriffe zu geben und eine Taxonomie dieser zu erstellen.
Dazu wird in Abschnitt 2 zunächst auf die grundlegenden Daten eingegangen und zusätzliche Fakten genannt.
In Abschnitt 3 wird aus untersuchten Vorfällen eine Taxonomie der Diebstähle erstellt.
Anschließend wird in Abschnitt 4 mit Hilfe dieser Taxonomie untersucht, wie sich die Angriffe entwickeln, das heißt, ob es aktuell oder in der Vergangenheit Trends gab oder immer wieder die gleichen Fehler gemacht wurden.
Aufgrund dieser Erkenntnisse werden dann in Abschnitt 5 grundlegende Verteidigungsstrategien vorgestellt.
In Abschnitt 6 wird die Auswirkung von Diebstählen auf die Verwendung auf Kryptowährungen diskutiert. In Abschnitt 7 wird ein Überblick über andere, verwandte Arbeiten gegeben.
Abschließend werden die Erkenntnisse des Papers in Abschnitt 8 noch einmal zusammengefasst dargestellt, die wichtigsten Erkenntnisse hervorgehoben und eine Einschätzung zur Zukunft im Bezug auf Kryptocoin Diebstähle gegeben.

2. DATEN

Die erfolgreichste Kryptowährung Bitcoin (BTC) existiert seit 2009 [4]. Erstmals gehandelt wird sie 2010. Der Preis für ein Bitcoin liegt in jenem Jahr bei maximal 0,30 USD [1]. Die ersten großen Bitcoindiebstähle finden 2011 statt.
Das Paper beschäftigt sich mit Fällen zwischen 2011 und 2017. Es werden für die Arbeit insgesamt 55 Diebstähle von verschiedenen Kryptotoken untersucht. Dabei werden nur Diebstähle an Services und nicht an Einzelpersonen betrachtet.
Der Gesamtwert der in den untersuchten Vorfällen gestohlenen Coins beträgt 1,4 Milliarden US-Dollar [1]. Den größten Anteil hat dabei der Hack der Coinhandelsplattform Coincheck. Bei diesem Hack wurden insgesamt 523 Millionen NEM mit einem damaligen Gegenwert von 523 Millionen US-Dollar gestohlen [20]. Zuvor war der Hack von MtGox, einer weiteren Coinbörse, mit Abstand der für die Angreifer ertragreichste. Im Fall MtGox wurden insgesamt wurden 650.000 Bitcoins mit damaligem Wert [2] von ca. 359 Millionen US-Dollar gestohlen [8].

2011	2012	2013	2014	2015	2016	2017	2018
7	8	12	4	4	5	13	2

Tabelle 1: Anzahl untersuchter Diebstähle pro Jahr

Der Großteil der untersuchten Vorfälle stammt aus den Jahren 2013 sowie 2017. Eine Übersicht über die Anzahl der Fälle pro Jahr ist in Tabelle 1 zu finden. Betrachtet wird hierbei jeweils das Datum der ersten vom Besitzer der Coins nicht autorisierten Überweisung. Dies ist insofern wichtig, da mehrere der Diebstähle jahresübergreifend stattfanden. Für Fälle vor 2014 wird [35] als Grundlage genutzt, da Aufgrund der mangelnden Interesse der Medien zu dieser Zeit wenige Quellen existieren. Die entnommenen Fälle und Quellen wurden zusätzlich geprüft.
Test

3. TAXONOMIE

Im folgenden Abschnitt werden die verschiedene Charakteristika anhand von Beispielen vorgestellt und definiert.

3.1 Art der Diebstähle

[1] Die gestohlenen Token werden jeweils mit dem zum Diebstahlzeitpunkt gültigen Kurs bewertet. Dazu werden aus den historischen Daten von `https://www.finanzen.net/devisen/bitcoin-dollar-kurs/historisch` der Abschlusskurs des jeweiligen Tages verwendet.
[2] Schlusskurs vom 24.2.2014: 552,85, erste offizielle Angabe des Verlustes

doi: 10.2313/NET-2018-03-1_06

Bei den verschiedenen Diebstählen kann zunächst zwischen unterschiedlichen Kategorien unterschieden werden, die wiederum eigene Unterarten aufweisen können. Die in den Vorfällen vorgekommenen Arten von Diebstählen werden im Folgenden vorgestellt, definiert und mit einem Beispiel erklärt.

3.1.1 Hacks

40 der 55 untersuchten Diebstähle wurden mithilfe von Hacking ausgeführt. Hacking bezeichnet hierbei das unerlaubte Eindringen in ein System bzw. das illegitime Nutzen eines Systems. Neben dem Fakt, dass Hacks die größte Gruppe der Diebstähle bilden, sind sie auch mit ca. 1,36 Milliarden US-Dollar für den größten Teil des gestohlenen Wertes verantwortlich. In 11 der Fälle wurde neben der Angabe, dass es einen Hack gab, keine weiteren Informationen veröffentlicht. Aufgrund der fehlenden Informationen wird oft auch ein Betrugsversuch vermutet, bei dem die Betreiber einen Hack nur vortäuschen, um so in den Besitz der von den Nutzern eingezahlten Kryptocoins zu kommen [3]. Dies trifft in 4 Fällen zu. Über diese Fälle sind nicht genug Informationen vorhanden um ein Urteil zu fällen.

Sicherheit des angegriffenen Services: Ein erstes Merkmal bei Hacking-Angriffen ist die Qualität der Sicherheitsmaßnahmen. Ein Beispiel für einen Fall, in dem keine Sicherheitsmaßnahmen genutzt wurden, ist der Hack von Sheepmarket. Sheepmarket war eine Darknet Drogenhandelsplattform. Laura Shin, Autorin beim Wirtschaftsmagazin Forbes, schreibt dazu in einem Artikel [24]: „There was no hacking. This was just a poorly created website that was just exploited by very young kids who immediately accepted responsibility for their actions “.

Professionalität der Angreifer: Ähnlich zur Sicherheit des Angegriffenen Services ist auch die Professionalität der Angreifer ein interessantes Merkmal von Diebstählen durch Hacking. Dabei wird ein Angreifer als professionell bezeichnet, wenn er gezielt und möglicherweise über längere Zeit einen Angriff auf einen Service durchführt und sich auch von technischen Hürden nicht aufhalten lässt, während ein nicht professioneller Angreifer einen zufällig im System gefundenen Fehler ausnutzt und/oder über kein großes technisches Wissen verfügt.
Ein Beispiel für einen besonders professionell durchgeführten Angriff ist der Fall Bitstamp [32], bei dem über längere Zeit mithilfe von Social Engineering, Spear Phishing und in ein Word Dokument eingebetteter Schadsoftware Zugriff auf die Server erlangt werden konnte.

3.1.2 Betrug

Von den 55 Fällen handelt es sich bei 13 um Diebstähle durch Betrug. Im Folgenden werden verschiedene Arten von Betrug im Zusammenhang mit Kryptowährungen erläutert.

Ponzi-Schema: Bei einem Ponzi-Schema wird mit hohen Renditen zum investieren angeregt. Die hohen Renditen werden dabei über neue Investoren gedeckt. Wenn der Zustrom der Investoren abbricht, werden zunächst meist technische Probleme vorgetäuscht, bis irgendwann die Seite, auf der investiert werden kann, verschwindet [2]. Ein Beispiel von vielen hierfür ist die Website Richmond Berks, die 1,4% Rendite pro Tag anbot, bevor sie verschwand [9]. Auch wenn in den hier untersuchten Fällen die Zahl dieser Betrugsfälle gering ist, ist diese insgesamt nicht zu unterschätzen. Eine Liste von Betrug im Zusammenhang mit Bitcoin lässt sich unter `http://www.badbitcoin.org/thebadlist/` finden. Oft werden Ponzi-Schemas von Betrügern als sogenannte High Yield Incestmens Plans (HYIP) angeboten [17]. Sie versprechen sehr hohe Renditen (mehrere Prozent pro Tag), wobei sie durch den Fachbegriff Vertrauen erzeugen wollen. HYIPs gibt es nicht nur im Bereich der Kryptowährungen. Sie sind allgemein ein beliebtes Betrugswerkzeug.

Exit Scam: Ein Exitscam bezeichnet einen Betrug, bei dem die Betreiber eines Services diesen abschaltet und dabei die Guthaben der Nutzer nicht auszahlt. Dies ist häufig bei illegalen Drogenmarktplätzen im Darknet der Fall. Der Service muss allerdings nicht immer illegal sein. Ein Beispiel hierfür ist die chinesische Bitcoinbörse GBL, die mit den bei ihr hinterlegten Bitcoins im Wert von 4,1 Millionen US-Dollar verschwand [23].

Fake Coin: Bei einem Fake-Coin-Betrug werden Schein-Kryptowährungen an Anleger verkauft. Ein Beispiel dafür ist die angebliche Währung E-Coin, mit der Betrüger sich insgesamt über 4 Millionen Schweizer Franken[4] aneignen konnten [30].

Phishing/Fake Apps: Beispielhaft für den Punkt Phishing/Fake Apps steht der aktuelle Fall Iotaseed (2018) [22]. Iotaseed war eine Applikation die es Nutzern der Cryptowährung IOTA ermöglichte sogenannte Seeds zu generieren. Diese Seeds lassen sich dabei mit einem Privatekey, der z.B. bei Bitcoin verwendet wird, vergleichen. Ne Der Service speicherte diese Seeds allerdings um später die mit den Seeds verwalteten IOTA Token zu stehlen.

3.2 Täter

Weiter soll bei den verschiedenen Diebstählen nach der Art des Täters unterschieden werden. Diese verschiedenen Täterprofile werden im folgenden definiert und erläutert.

3.2.1 Hacker

Ein Großteil der Täter lässt sich als Hacker beschreiben. Dabei bezeichnet der Begriff Hacker, analog zum Begriff Hack in Unterunterabschnitt 3.1.1, Angreifer die unerlaubt in ein System eindringen oder es auf illegitime Weise nutzten. Durch die große Anzahl ist eine weitere Differenzierung interessant, aber aufgrund von wenig Informationen nicht immer unproblematisch.

Professionelle (nichtstaatliche) Hacker (Black Hats): Als professionelle Hacker werden Hacker eingeordnet, die wie im Fall Bitstamp gezielt und mit technisch versierten Mitteln Angriffe ausführen. Zum Fall Bitstamp ist dabei ein interner Bericht geleakt worden, der interessante Einblicke in einen solchen Angriff gibt [32].

Staatliche Hacker: Es gibt noch keinen Fall in dem bewiesen wurde, dass staatliche Hacker für einen Diebstahl

[3] Solange es allerdings keine Beweise dafür gibt werden die Vorfälle als Hack bewertet.

[4] ca. 4 Millionen US-Dollar

doi: 10.2313/NET-2018-03-1_06

verantwortlich waren. Im Fall Bitthumb, der größten Südkoreanischen Coinbörse, wird jedoch von den Behörden ein Angriff von Nordkorea bzw. nordkoreanischen Hackern vermutet [12]. Als staatliche Hacker sind Hacker, die ihre Fähigkeiten für einen Staat einsetzen. Sie müssen dabei jedoch nicht offiziell für den Staat arbeiten, sondern können auch nur geduldet sein.

White Hat Hacker: White Hat Hacker versuchen durch ihre Fähigkeiten Sicherheitslücken zu entdecken, bevor sie von kriminellen Hackern ausgenutzt werden können. Beispiel hierfür ist dabei der Hack des DAO Projektes[5]. Das DAO Projekt war der Versuch über Ethereum Smart Contracts eine dezentralisierte autonome Organisation zu erschaffen. Problematisch war allerdings, dass die Funktion, die es Investoren erlaubte, ihre Investition aus the DAO zu entziehen, einen Fehler hatte. Dem Angreifer war es durch diesen Fehler möglich, die Auszahlungsfunktion innerhalb der Funktion erneut aufzurufen und somit die Überprüfung der Legitimität der Auszahlung zu umgehen [28]. Allerdings konnten White Hat Hacker, nachdem die Sicherheitslücke bekannt wurde rechtzeitig die in das DAO Projekt eingezahlte Token sichern [26].

3.2.2 Betrüger

Analog zu Unterunterabschnitt 3.1.2 gibt es die Täterkategorie Betrüger. Große Unterscheidungen gibt es bei diesem Typ nicht. Möglicherweise kann man in Zukunft Wiederholungstäter identifizieren.

3.2.3 Mitarbeiter

Die Täterkategorie Mitarbeiter wird im Fall Shapeshift sinnvoll. In diesem verkaufte ein Mitarbeiter Informationen und Zugangsdaten an eine Person, die daraufhin mehrmals in das System von Shapeshift einbrach [31]. Genauere Informationen zu diesem Vorgang enthält eine auf Wunsch von Shapeshift veröffentlichte Analyse [6].

3.3 Opfer der Diebstähle

Die Kategorie Opfer der Diebstähle beschäftigt sich mit der Frage wer letztendlich den Schaden trägt.

3.3.1 Service(-betreiber):

Bei seriösen Services wird oft versucht, den Schaden der Nutzer zu übernehmen. Allerdings kann es dabei dazu kommen, dass dies über einen längeren Zeitraum geschieht, wenn die gehackten Services nicht genügend Reserven haben. Ein Modell für eine solche Lösung wäre die Ausgabe eines neuen Token, der nach und nach von der Börse wieder aufgekauft wird.

Coinbörsen: Coinbörsen sind in den untersuchten Fällen am häufigsten (insgesamt 21 mal) das Opfer von Diebstählen geworden. Coinbörsen sind Handelsplätze für Kryptowährungen, ähnlich wie eine Börse.

Miningpools: Im Zusammenhang mit Kryptowährungen ist auch das sogenannte minen (deutsch. Abbauen) zu nennen. Um eine Transaktion in einer Blockchain zu übernehmen muss meist eine aufwendige Berechnung ausgeführt werden [7], deren Ergebnis wiederum einfach zu überprüfen ist. Der Versuch, diese Aufgabe zu lösen, wird mining genannt. Derjenige, der die Lösung als erstes berechnet, erhält dafür eine Belohnung in der jeweiligen Währung der Blockchain. Detaillierter erklärt wird dies unter anderem in einem Artikel auf BTC Echo[13].
Miningpools sind Zusammenschlüsse mehrerer Personen bzw. Hardwaresysteme. Durch den Zusammenschluss wird eine größere Rechenleistung und damit eine höhere Wahrscheinlichkeit, eines dieser Probleme zu lösen, erreicht. Die Belohnung wird dabei gerecht anhand der eingebrachten Rechenleistung verteilt. Dazu werden allerdings die geminten Token in einem zentralen Wallet zwischengespeichert. Oft wird auch erst auf Wunsch ausgezahlt. Somit ist dieses Wallet ein lohnendes Ziel für Angreifer.
Von den untersuchten Fällen waren 4 Angriffe auf Miningpools.

Webwallets: Webwallets sind Serviceanbieter die Nutzern ermöglichen mit den verschiedenen Blockchains zu interagieren, ohne selbst eine Software dafür installieren zu müssen. Dabei ist darauf zu achten, wie der private Schlüssel des Wallets gesichert wird. Allgemein ist zu bedenken, dass ein solcher Anbieter ein deutlich lukrativeres Ziel für Hacker ist als ein einzelnes lokales Wallet.

Startups: Der innovative Charakter der Blockchain begünstigt die Gründung von Startups. Kryptocoinbezogene Startups versuchen sich dabei meist über ein sogennantes Initial Coin Offering, wie unter Unterunterabschnitt 3.4.3 beschrieben, zu finanzieren.

Darknet Markt: Der Begriff Darknet bezeichnet ein Netzwerk, das innerhalb des Internets existiert, aber zusätzliche Programme mit zusätzlichen Protokollen benötigt, um darauf zuzugreifen. Dabei wird durch diese versucht, die Identität der Kommunikationspartner geheimzuhalten. Aus diesem Grund wird das Darknet auch oft verwendet, um illegalen Aktivitäten wie dem Handel mit Drogen nachzugehen. Dies geschieht dafür auf eigens geschaffenen Plattformen. Diese Anonymität führt allerdings oft auch zu Betrugsfällen. Des weiteren ist das Sicherheitsniveau dieser Seiten oft nicht so hoch wie das von legalen Handelsplätzen außerhalb des Darknets. Ein Beispiel für einen unsicheren gehackten Darknethandelsplatz ist Sheepmarket [24].

Lokale Clients: Ein Blockchainclient, oft fälschlicherweise Wallet genannt, ist ein Programm, das dabei hilft, Kryptowährungen zu verwalten und mit der dazugehörigen Blockchain zu interagieren. Da diese dadurch auch mit dem Internet verbunden sind, können auch sie Ziel eines Angriffs sein. Als Beispiel dient hier der Ethereum-Client *Parity*. Dieser hatte einen Fehler in der Implementierung von sogenannten Multisignature Wallets. Ein Multisignature Wallet kann man sich als ein Konto vorstellen, das bei Überweisungen die Autorisierung durch die Mehrheit der Besitzer erfordert. Durch einen Bug in der Implementierung konnte die Besitzerliste eines solchen Wallets überschrieben und somit die darin enthaltenen Token gestohlen werden [34].

[5] DAO steht für dezentrale autonome Organisation

[6] `https://de.scribd.com/doc/309591980/ShapeShift-Postmortem`

[7] Oft wird mit einer Berechnung gleich ein ganzer Block validiert

doi: 10.2313/NET-2018-03-1_06

Shopping: Durch die extremen Schwankungen der Preise von Kryptowährungen sind Händler, die solche akzeptieren, selten. Sollte sich jedoch eine Kryptowährung als Zahlungsmittel etablieren, könnte diese Kategorie häufiger vertreten sein.
Unter den untersuchten Fällen befindet sich mit Purse.io allerdings nur ein Anbieter, der Einkäufe bei Amazon mit Bitcoin ermöglichen will.

3.3.2 Kunden/Nutzer:
Vor allem zu Beginn der Kryptowährungsära (2011-2013) trugen letztendlich die Nutzer den Schaden davon.

3.3.3 Investoren:
Durch die teils extremen Wertsteigerungen von verschiedenen Kryptowährungen sind Kryptowährungen mittlerweile auch für Investoren interessant. Diese können allerdings nicht nur durch die extremen Kursschwankungen Verluste machen, sondern auch durch Betrug oder den Hack eines von ihnen unterstützten Startups.

3.4 Angriffziel/Angriffsfläche
Im folgenden wird aufgearbeitet, über welche Angriffsflächen Services angegriffen werden.

3.4.1 Server/Software
Die Server eines Services bzw. Software, die im Zusammenhang mit Kryptocoins verwendet wird, sind eine klassische Angriffsfläche. Dabei deckt diese Kategorie sowohl die Software der Services als auch die der Blockchain und darauf basierender Technologien ab.

Smart Contract Ein Smart Contract ist Code, der in einer Blockchain ausgeführt wird wenn gewisse Bedingungen eintreten [6]. Dadurch lassen sich unter anderem Multisignaturwallets, Verträge oder sogar ganze Organisationen realisieren. Trotz allem sind Smart Contracts Code, so dass es möglich ist Fehler in diesem auszunutzen. Das bedeutet im Zusammenhang mit Blockchains und Kryptowährungen, dass diese Fehler direkt zu einem finanziellen Schaden führen können. Beispiele für Angriffe auf Smart Contracts sind das Multisignaturewallet des Ethereumclients *Parity* [34] oder das DAO Projekt, dass bereits in Unterunterabschnitt 3.2.1 angesprochen wurde.

Coinprotokoll Beim Zieltyp Coinprotokoll geht es um Angriffe auf die Protokolle der verschiedenen Blockchains, wie der theoretisch mögliche 51% Angriff [25], der allerdings ein allgemeines Problem von konsensbasierten P2P Netzwerken ist. Bisher kam es ein einziges Mal zu einem Problem mit dem Protokoll von Bitcoin. 2010 wurden durch einen "value overflow"knapp 185 Milliarden Bitcoins erzeugt. Die Anzahl der Bitcoins ist eigentlich auf 21 Millionen begrenzt. Durch einen „soft-fork“ konnte die Erzeugung der neuen Bitcoins annulliert werden [3].

3.4.2 Webpräsenz/Social Media
Der Punkt Webpräsenz überschneidet sich auf den ersten Blick mit dem vorhergegangenen Punkt Server/Software. Webpräsenz/Social Media bezieht sich allerdings nicht darauf, diese über Sicherheitslücken anzugreifen, sondern durch die Übernahme dieser, falsche Informationen an die Nutzer/Investoren weiterzugeben. Zu beobachten war diese Angriffsfläche meist im Zusammenhang mit Startups.

3.4.3 Initial Coin Offerings
Initial Coin Offerings (ICOs) sind eine Art des Crowdfundings von Kryptowährungsstartups. Dabei erhalten Investoren gegen herkömmliche oder kryptobasierte Währung Token des neuen Unternehmens. Dabei ist das Initial Coin Offering die erste Möglichkeit an diese Token zu kommen. ICOs sind ein relativ neues Modell. Dies sieht man auch daran, dass die vier Angriffe auf ICOs alle im Jahr 2017 stattfanden. Die Angriffe auf ICOs stehen oft im engen Zusammenhang mit Attacken auf die Webpräsenz oder die Social Media Kanäle des Startups, wie die Beispiele CoinDash [7] und Enigma.co [18] zeigen.

3.4.4 Mitarbeiter
Neben Angriffen auf Software sind Angriffe über die Mitarbeiter eines Unternehmens ein möglicher Weg für Angreifer. Ein Szenario dabei ist ein Angriff über Spear Phishing, wie unter Unterabschnitt 3.5 beschrieben, bei dem die Mitarbeiter dazu gebracht werden sollen, dem Angreifer unwissend Zugang zu den Systemen zu ermöglichen. Allerdings können die Mitarbeiter nicht nur das Angriffsziel sein, sondern auch die Täter wie der Fall Shapeshift zeigt [31].

3.4.5 Webhoster
Mit Linode wurde auch ein Webhoster im Zusammenhang mit Kryptowährungen angegriffen [15]. Auf Linode wurden die Wallets mehrerer Coinbörsen gehostet, die über diesen Weg gelehrt werden konnten. Allgemein ist der Hack des Webhosters nicht nur für Coinbörsen gefährlich sondern für jeden Service der keine eigenen Server verwendet.

3.5 Angriffsmethode
Neben der Angriffsfläche ist auch die genutzte Angriffsmethode interessant. Dabei hängen manche Angriffsflächen und Methoden, wie Software und Exploits, eng zusammen. Im Folgenden werden verschiedene, in den Fällen vorgekommenen Angriffsmethoden vorgestellt.

Exploits: Der Begriff Exploit bezeichnet Code, der gezielt Schwachstellen in anderem, fremden Code ausnutzt, um in diesem eigenen Code auszuführen oder ein ungewünschtes Verhalten herbeizuführen. Beispielhaft das Überweisen von Kryptowährung ohne das Wissen oder die Autorisierung des Besitzers. Ein Fall in dem ein, wenn auch relativ einfacher, Exploit genutzt ist ist der Hack von Sheepmarket [24].

Spearphishing: Bei Spearphishing handelt es sich um einen gezielten Angriff auf einige wenige Personen. Dabei werden auf die Interessen der Personen abgestimmte Nachrichten verfasst, die sie letztendlich zum ausführen von Code oder der Herausgabe von Informationen bringen soll. Ein solcher Angriff kann sich auch über eine längere Zeit ziehen. Beispielhaft dazu ist der Hack der Coinbörse Bitstamp zu nennen [32].

Defacing von Webseiten Das Defacing von Webseiten beschreibt einen Angriff bei dem zunächst Zugriff auf den Webserver des anzugreifenden Unternehmens erlangt wird. Im allge-

meinen werden dann Informationen auf der Seite geändert. Im Bezug auf Kryptowährungen werden dann vor allem Informationen wie Adressen für Spenden oder Investitionen geändert. Ein mögliches Fallbeispiel ist der Hack des Initial Coin Offerings CoinDash. Dort wurde die Adresse, an die Investoren Ethereum senden sollten, geändert. Die Angreifer bekamen damit Ether im damaligen Gesamtwert von 7,6 Millionen US-Dollar von den Investoren überwiesen [7].

Spam/Allgemeines Phishing: Im Gegensatz zu Spearphishing zielt allgemeines Phishing auf die breite Masse ab. Dazu können im Idealfall (aus Sicht der Angreifer) gehackte Social Media Accounts und E-Mail Adressen verwendet werden. Ein Beispiel hierfür ist der Angriff auf das ICO Enigma.co [18].

Inside Job: Als Inside Job wird ein Angriff genannt, der von Innen durch einen Mitarbeiter oder mit Hilfe eines Mitarbeiters durchgeführt wird. Im Fall Shapeshift wurde angestoßen durch einen Mitarbeiter insgesamt drei mal das Hot Wallet gelehrt und letztendlich Coins im Wert von 230.000 US-Dollar entwendet [33].

3.6 Art der Coins

Unterschieden werden kann auch zwischen den verschiedenen Arten von Coins die gestohlen wurden.

Bitcoin (BTC): Bitcoin ist die erste Kryptowährung und existiert seit 2009. Zuerst gehandelt wurde Bitcoin im Jahre 2010 [10]. Aufgrund des großen Wertes ist Bitcoin im Vergleich zu anderen Kryptowährungen am häufigsten Ziel von Diebstählen. Das belegt auch, dass in 42 der untersuchten Fällen Bitcoins gestohlen wurden. In zwei davon wurden neben Bitcoin auch anderen Kryptowährungen gestohlen, dennoch machte Bitcoin den größten Anteil aus.

Ether (ETH): In 10 der untersuchten Fälle wurde Ether (ETH) gestohlen. Ether ist der Token, der im Ethereumnetzwerk genutzt wird. Vorteil von Ethereum ist, dass es im Gegensatz zu Bitcoin nicht nur als Währung genutzt werden kann. Vielmehr bietet Ethereum die Möglichkeit sogenannter Smart Contracts. Ein Smart Contract ist Code der im Ethereumnetzwerk ausgeführt wird [6]. Ether besitzt, nach Bitcoin, die zweitgrößte Marktkapitalisierung [14].

Sonstige: Neben Bitcoin und Ether gibt es noch weitere Coins, wie z.B. Dash, Litecoin, Monero oder Tether. Dabei ist zu beachten, dass manche Blockchains bzw. Coins voneinander abstammen. Dies kann dadurch geschehen, dass sich die Blockchain aufgrund von Differenzen in der jeweiligen Community aufspaltet[8]. Beispiele dafür wären Ethereum und Ethereum Classic sowie Bitcoin und Bitcoin Cash. Eine andere Möglichkeit ist, dass eine neue Blockchain aufbauend auf dem Code einer bereits existierenden Blockchain geschaffen wird.

3.7 Schadenshöhe und Auswirkung

Insgesamt wurden über 1,4 Milliarden US-Dollar in den untersuchten Fällen gestohlen. 1,36 Milliarden US-Dollar davon durch Hacking, 6 Millionen US-Dollar durch Betrug.

[8]Der Fachbegriff für ein solches aufspalten ist Fork

Verwendet man den heutigen Kurs[9] zu Berechnung des Wertes, wurden insgesamt mindestens 22 Milliarden US-Dollar in Bitcoins (1259185 BTC) gestohlen.
Der finanzielle Rahmen spannt sich dabei für Hacks von 15.534 US-Dollar, im Falle des Hacks des Glückspielanbieters betco.in, bis zu um die 534 Millionen US-Dollar im Fall Coincheck. Im Bezug auf Betrug wurden in den untersuchten Fällen, in denen Angaben zur Höhe gemacht wurden, zwischen 6461 US-Dollar und 4,3 Millionen US-Dollar gestohlen. Bei Betrugsfällen ist die Datenlage jedoch schwieriger, so dass oft gar keine Informationen über die Menge an gestohlenen Coins bzw. Geld verfügbar ist.
Den Großteil des gestohlenen Geldes macht das Geld aus Coinbörsen aus (1,1 Milliarden US-Dollar).
Der Hack der Coinbörse Cointrader macht dabei mit 33.600 US-Dollar den kleinsten Anteil aus.

Die fünf Diebstähle mit dem größten gestohlenen Wert sind:

1. Coincheck, 534 Millionen US-Dollar, Coinbörse, gehackt
2. MtGox, 359 Millionen US-Dollar, Coinbörse, gehackt
3. Bitthumb, 82 Millionen US-Dollar, Coinbörse, gehackt
4. Bitfinex, 71 Millionen US-Dollar, Coinbörse, gehackt
5. DAO, 62 Millionen US-Dollar, Smart-Contract im Ethereum Netzwerk, gehackt

4. (ZEITLICHE) KORRELATION DER CHARAKTERISTIKA

Kryptowährungen existieren seit 8 Jahren. Im folgenden wird deshalb untersucht, wie sich Angriffe seitdem verändert haben.

Die ersten Angriffe 2011 waren meist geringer Sicherheit geschuldet. Während es bei Angriffen auf MtGox (2011) und Sheepmarket (2013) nach Angaben noch (fast) keine Sicherheitsmaßnahmen gab [24], die umgangen werden mussten, werden heute für Angriffe deutlich besser Fähigkeiten benötigt. Die gestiegenen Kurse haben jedoch dazu geführt, dass Hacker mit entsprechenden Fähigkeiten mittlerweile Kryptowährungen als lohnendes Ziel ansehen.

Im Bezug auf den gestohlenen Gegenwert hat sich jedoch keine Entwicklung gegeben. Das heißt es gibt immer noch Fälle in denen sehr große Beträge im Millionenbereich gestohlen werden. Dies hängt aber wiederum mit den extrem gestiegenen Wechselkursen zusammen.

Coinbörsen sind das klassische Angriffsziel. In allen Jahren von 2011 bis 2017 gab es mindestens einen Hack einfacher Coinbörse. Im Allgemeinen lässt sich trotzdem ein Anstieg der Sicherheitsvorkehrungen nachvollziehen. So wird mittlerweile von nahezu allen Coinbörsen
2-Faktor-Authentifizierung unterstüzt und die Mittel in Hot und Cold Wallets aufgeteilt.

Mit dem Aufkommen von Initial Coin Offerings im Jahr 2017 hat es auch verstärkt Angriffe auf diese gegeben. Dabei

[9]Kurs am 20.12.2017 um 12:50 : 17321,97 US-Dollar pro Bitcoin

lässt sich feststellen, dass häufig die offiziellen, öffentlichen Kanäle der Startups von Angreifern übernommen werden. Die Überprüfung, ob eine Blockchainadresse wirklich einem Startup, Service oder Endanwender gehört, könnte in Zukunft ein interessantes Problem werden, da es für Statups und Services aus Sicherheitsgründen sehr wichtig ist, wohingegen einige Nutzer möglicherweise Kryptowährungen auch wegen der teilweise gegebenen Anonymität nutzten und diese bewahren wollen.

Gesteigert haben sich vor allem die Anzahl der Betrugsfälle im Bezug auf Kryptowährungen. Dies lässt sich darauf zurückführen, dass zum einen der Preis von Kryptowährungen mittlerweile sehr hoch ist und zum anderen immer mehr Personen Kryptowährungen verwenden, die wenig Wissen über Technologie und Gefahren besitzen. Die meisten Betrugsfälle lassen sich bis auf den Umstand, dass sie den aktuellen Hype um Kryptowährungen ausnutzen, nicht von Betrugsmethoden in anderen Bereichen unterscheiden. Sie sind also im Gegensatz zur innovativen Technologie Blockchain konservativ.

In Zukunft ist vor allem abzuwarten, welche neuen Dienste welche neuen Angriffsflächen bieten. Dabei könnten neben klassischen Services wie zum Beispiel Shopping oder Glücksspielwebseiten auch innovativere Dienste mit völlig neuen Schwachstellen auftreten.

5. VERTEIDIGUNGSSTRATEGIEN

Im Folgeden sollen grundlegende Verteidigungsstrategien für die verschiedenen Opfertypen gegeben und auf weiterführendes Material verwiesen werden.

5.1 ... für Nutzer

Kryptowährungen sind ein Zahlungsmittel. Deshalb sollten sie auch wie herkömmliche Zahlungsmittel behandelt werden. Das bedeutet, dass Nutzer im übertragenen Sinne nicht ihr ganzes Geld bei sich haben sollten. Auf Kryptowährungen übertragen bedeutet das, dass es sinnvoll ist zwischen einem sogenannte Hot-Wallet und einem Cold-Wallet zu unterscheiden. Ein Hot-Wallet wird dabei im Alltag genutzt und sollte nicht mehr Coins enthalten als benötigt. Ein Cold-Wallet wiederum wird offline auf einer Festplatte oder einem nicht mit dem Internet verbundenen Rechner gespeichert. Es besteht auch die Möglichkeit eines sogenannten Paperwallets. Bei diesem werden Public- und Privatekey in einem maschinenlesbaren Format auf ein Stück Papier gedruckt. Auch wichtig ist es Backups des Wallets zu machen. Ohne den dazugehörigen Privatekey gibt es keine Möglichkeit über die eigenen Kryptotoken zu verfügen. Immer zu bedenken ist, dass Überweisungen in den verschiedenen Blockchainnetzwerken nicht rückgängig gemacht werden können.

5.2 ... für Servicebetreiber

Servicebetreiber sollten sich bewusst sein, dass sie oft einen bankenähnlichen Status einnehmen und somit Verantwortung für die Sicherheit der eingezahlten Coins haben. Wie für den einzelnen Nutzer ist auch Services die Nutzung von Hot- und Cold-Wallets zu empfehlen. Zusätzlich können auch Hardware-Sicherheitsmodule eingesetzt werden. Wie ein Einsatz eines solchen Moduls aussehen könnte wurde unter anderem unter [11] diskutiert. Es sollte darauf geachtet werden, dass Mitarbeiter starke Passwörter, die regelmäßig geändert werden, und 2-Faktor-Autorisierung verwenden. Negativbeispiel ist hierbei das ICO Startup Enigma, dessen CEO ein Passwort verwendete, dass, zusammen mit seiner E-Mail Adresse, bereits in der Vergangenheit durch den Hack von anderen Services geleakt wurde [18].

5.3 ... für Investoren

Kryptowährungen werden mittlerweile oft als Investition gesehen. Dabei ist zu bedenken, das die meisten Kryptowährungen extremen Wertschwankungen unterliegen. Von Investitionen die hohe Renditen ohne Risiko versprechen (HYIPs) sollte Abstand gehalten werden. Wchtig ist auch, sich ausreichend über die verschiedenen Währungen und Technologien zu informieren.

6. AUSWIRKUNGEN AUF DIE VERWENDUNG VON KRYPTOWÄHRUNGEN

Nach großen Hacks wurde immer wieder ein Ende der Kryptowährungen vorausgesagt [19]. Zwar gab es nach größeren Angriffen immer wieder Kurseinbrüche. Die Kurse der verschiedenen Kryptowährungen konnten sich bisher aber immer erholen.

Besonders hervorzuheben ist auch, dass es, bis auf die durch einen Value Overflow erzeugten zusätzlichen Bitcoins (siehe Unterunterabschnitt 3.4.1), keine Verluste\Diebstähle durch Fehler in den verschiedenen Blockchains gab. Weitaus kritischer für den Erfolg von Blockchain-basierten Währungen könnte andere Faktoren sein. Ein Punkt ist zum Beispiel Probleme wie der sehr hohe Stromverbrauch von Bitcoin[29].

7. VERWANDTE ARBEITEN

Neben dem allgemeinen Hype um Kryptowährungen wie Bitcoin sind diese und die dazugehörigen Technologien auch in der Informatik ein sehr aktuelles Thema. Das Paper Beware the Middleman: Empirical Analysis of Bitcoin-Exchange Risk [27] untersuchte 40 verschiedene Coinbörsen. Dabei lag der Fokus auf der statistischen Auswertung. Unter anderem wurde untersucht, bei welchen Eigenschaften von Coinbörsen die Wahrscheinlichkeit der Schließung am höchsten ist. Ein anderes Paper [5] beschäftigt sich mit den Gefahren einer blockchainbasierten Währung, vor allem im Bezug auf kriminelle Machenschaften wie Geldwäsche und Darknet Drogenmarktplätze. Eine weitere Arbeit [16] beschäftigen sich mit der möglichen Bedrohung herkömmlicher Währungen und Finanzinstituten durch Bitcoin und inwiefern eine Regulierung notwendig ist.

8. ZUSAMMENFASSUNG

In diesem Paper wurden anhand verschiedener Diebstähle von Kryptowährungen Charakteristika solcher herausgearbeitet. Dazu wurden zunächst grundlegende Daten dargestellt. Anschließend wurden die verschiedenen Teile der Taxonomie definiert und mit Hilfe von Beispielen erläutert. Es wurde auf die Entwicklung der Charakteristika eingegangen, einige Verteidigungsstrategien angesprochen. Abschließend wurde Zusammengefasst, welche Auswirkungen die Diebstähle auf die verschiedenen Kryptowährungen hatten.

9. LITERATUR

[1] Bitcoin Price History. https://99bitcoins.com/price-chart-history/. Zuletzt aufgerufen am 14.12.2017.

[2] Ponzi Scheme. https://www.investor.gov/protect-your-investments/fraud/types-fraud/ponzi-scheme. Zuletzt aufgerufen am 24.01.2018.

[3] . Value overflow incident. https://en.bitcoin.it/wiki/Value_overflow_incident, 2014. Zuletzt aufgerufen am 20.12.2017.

[4] Genesis block. https://en.bitcoin.it/wiki/Genesis_block, 3.2.2009. Zuletzt aufgerufen am 15.12.2017.

[5] S. T. Ali, D. Clarke, and P. McCorry. *Bitcoin: Perils of an Unregulated Global P2P Currency*, pages 283–293. Springer International Publishing, Cham, 2015.

[6] Alyssa Hertig. How Do Ethereum Smart Contracts Work? https://www.coindesk.com/information/ethereum-smart-contracts-work/. Zuletzt aufgerufen am 19.12.2017.

[7] Axel Kannenberg. Coindash: Kryptogeld-Crowdfunding mit Adressänderung gekapert. https://www.heise.de/newsticker/meldung/Coindash-Kryptogeld-Crowdfunding-mit-Adressaenderung-gekapert-3778227.html, 2017. Zuletzt aufgerufen am 16.12.2017.

[8] Axel Kannenberg. Geld gewaschen und Mt. Gox bestohlen: Mutmaßlicher Chef der Bitcoin-Börse Btc-e verhaftet Update. https://www.heise.de/newsticker/meldung/Geld-gewaschen-und-Mt-Gox-bestohlen-Mutmasslicher-Chef-der-Bitcoin-Boerse-Btc-e-verhaftet-3784467.html?artikelseite=2, 2017. Zuletzt aufgerufen am 15.12.2017.

[9] Bassam Jamal. Richmond Berks Review: Real Estate Ponzi Scam Stops Paying. http://binaryscamwatchmonitor.com/richmond-berks-review-scam/, 2017. Zuetzt aufgerufen am 16.12.2017.

[10] Bernard Marr. A Short History Of Bitcoin And Crypto Currency Everyone Should Read. https://www.forbes.com/sites/bernardmarr/2017/12/06/a-short-history-of-bitcoin-and-crypto-currency-everyone-should-read/#4d8a68623f27, 2017. Zuletzt aufgerufen am 20.12.2017.

[11] Bitexperts. Bitcoin HSM implementation guide for securing a server-side hosted wallet. http://bitexperts.com/Question/Detail/1142/bitcoin-hsm-implementation-guide-for-securing-a-server-side-hosted-wallet. Zuletzt aufgerufen am 20.12.2017.

[12] British Broadcasting Corporation (BBC). North Korea 'hacked crypto-currency exchange in South'. http://www.bbc.com/news/world-asia-42378638, 2017. Zuetzt aufgerufen am 18.12.2017.

[13] BTC-Echo. Wie funktioniert Bitcoin-Mining? https://www.btc-echo.de/tutorial/wie-kann-ich-bitcoins-minen/. Zuletzt aufgerufen am 24.01.2018.

[14] coinmarketcap.com. Cryptocurrency Market Capitalizations. https://coinmarketcap.com/. Zuletzt aufgerufen am 20.12.2017.

[15] Dan Goodin. Bitcoins worth $ 228,000 stolen from customers of hacked Webhost. https://arstechnica.com/information-technology/2012/03/bitcoins-worth-228000-stolen-from-customers-of-hacked-webhost/, 2012. Zuletzt aufgerufen am 16.12.2017.

[16] P. D. Filippi. *Bitcoin: A Regulatory Nightmare to a Libertarian Dream*. 2014.

[17] Financial Industry Regulatory Authority. HYIPs—High Yield Investment Programs Are Hazardous to Your Investment Portfolio. https://www.finra.org/investors/alerts/hyips-high-yield-investment-programs-are-hazardous-your-investment-portfolio, 2010. Zuletzt aufgerufen am 24.01.2018.

[18] Francisco Memoria. Hacker Nets over $500,000 after Hacking Enigma before ICO Date. https://www.ccn.com/hacker-nets-over-500000-after-hacking-enigma-before-its-ico-date/, 2017. Zuletzt aufgerufen am 21.12.2017.

[19] Jack Tatar. Is This the End of Bitcoin (Again)? https://www.thebalance.com/is-this-the-end-of-bitcoin-again-391268, 2015. Zuletzt aufgerufen am 20.12.2017.

[20] Jon Buck. Coincheck: Stolen $534 Mln NEM Were Stored On Low Security Hot Wallet. https://cointelegraph.com/news/coincheck-stolen-534-mln-nem-were-stored-on-low-security-hot-wallet, 2018. Zuletzt aufgerufen am 10.02.2018.

[21] Jörn Brien. Kursexplosion bei Bitcoin, Ether und Ripple: Börsen Coinbase und Bitfinex brechen zusammen. http://t3n.de/news/bitcoin-boersen-brechen-zusammen-885612/, 13.12.2017. Zuletzt aufgerufen am 13.12.2017, 10.54 Uhr.

[22] JP Buntinx. Online IOTA Seed Generator Starts Stealing Funds From Users. https://themerkle.com/online-iota-seed-generator-starts-stealing-funds-from-users/, 2018. Zuletzt aufgerufen am 10.02.2018.

[23] Kadhim Shubber. $4.1 Million goes missing as Chinese bitcoin trading platform GBL vanishes. https://www.coindesk.com/4-1m-goes-missing-chinese-bitcoin-trading-platform-gbl-vanishes/, 2013. Zuetzt aufgerufen am 16.12.2017.

[24] Laura Shin. Mystery Solved: $6.6 Million Bitcoin Theft That Brought Down Dark Web Site Tied To 2 Florida Men. https://www.forbes.com/sites/laurashin/2016/05/30/mystery-solved-6-6-million-bitcoin-theft-that-brought-down-dark-web-site-tied-to-2-florida-men/#34abc6f323d5, 2016. Zuletzt aufgerufen am 16.12.2017.

[25] learncryptography.com. 51% Attack. https://learncryptography.com/cryptocurrency/51-attack. Zuletzt aufgerufen am 20.12.2017.

[26] Lefteris Karapetsas. White Hat Siphoning has Occurred. What Now? https://blog.slock.it/white-hat-siphoning-has-occurred-what-now-f7ba2f8d20ef, 2016. Zuletzt aufgerufen am 18.12.2017.

[27] T. Moore and N. Christin. *Beware the Middleman: Empirical Analysis of Bitcoin-Exchange Risk*. Springer

Berlin Heidelberg, Berlin, Heidelberg, 2013.

[28] Phil Daian. Analysis of the DAO exploit. http://hackingdistributed.com/2016/06/18/analysis-of-the-dao-exploit/, 2016. Zuletzt aufgerufen am 24.01.2018.

[29] Powercompare.co.uk. Bitcoin Mining Now Consuming More Electricity Than 159 Countries Including Ireland & Most Countries In Africa. https://powercompare.co.uk/bitcoin/, 2017. Zuletzt aufgerufen am 24.01.2018.

[30] Reuters. Schweizer Behörden gehen gegen E-Coin-Betrüger vor. https://www.handelsblatt.com/finanzen/maerkte/devisen-rohstoffe/erfundene-kryptowaehrung-schweizer-behoerden-gehen-gegen-e-coin-betrueger-vor/20346512.html, 2016. Zuletzt aufgerufen am 18.12.2017.

[31] Shapeshift.io Blog. A Timeline: ShapeShift Hacking Incident. https://info.shapeshift.io/blog/2016/04/19/timeline-shapeshift-hacking-incident, 2016. Zuetzt aufgerufen am 16.12.2017.

[32] Stan Higgins. Details of $5 Million Bitstamp Hack Revealed. https://www.coindesk.com/unconfirmed-report-5-million-bitstamp-bitcoin-exchange/, 2015. Zuetzt aufgerufen am 16.12.2017.

[33] Stan Higgins. ShapeShift lost $230k in String of Thefts, Report finds. https://www.coindesk.com/digital-currency-exchange-shapeshift-says-lost-230k-3-separate-hacks/, 2016. Zuletzt aufgerufen am 24.01.2018.

[34] steemit.com. Parity Wallet Hack Explained. https://steemit.com/cryptocurrency/@etheraveum/parity-wallet-hack-explained, 2017. Zuletzt aufgerufen am 20.12.2017.

[35] User dree12. List of Major Bitcoin Heists, Thefts, Hacks, Scams, and Losses. https://bitcointalk.org/index.php?topic=576337, 2014. Zuletzt aufgerufen am 24.01.2018.

doi: 10.2313/NET-2018-03-1_06

A. LISTE VON UNTERSUCHTEN HACKS

Fall	Kategorie	Jahr	Anzahl Coins	Cointyp	Damaliger Wechselkurs	Höhe $ damals
MtGox	Coinbörse	2011-2014	650000	BTC	552,85 $/฿	359 Mio.
MyBitcoin	Coinbörse (Betrugsverdacht)	2011	78740	BTC	11,62 $/฿	10,73 Mio.
Bitomat	Coinbörse	2011	17000	BTC	13,62 $/฿	231.570
Bitcoin7	Coinbörse (Betrugsverdacht)	2011	5000	BTC	3,20 $/฿	15.980
Bitfloor	Coinbörse	2012	24086	BTC	11,34 $/฿	273.209
Linode	Webhoster	2012	46703	BTC	4,90 $/฿	228.845
Bitcoinia	Coinbörse	2012	38527	BTC	4,97 $/฿	191.638
Kronos	Startup	2012	4000	BTC	10,71 $/฿	42.859
BTC-E	Coinbörse	2012	4500	BTC	7,88 $/฿	35.452
Betcoin	Glücksspiel	2012	2900	BTC	5,36 $/฿	15.534
50BTC	Miningpool	2012	1174	BTC	11,45 $/฿	13.437
Silkroad	Darknetmarktplatz	2013	27618	BTC	752,50 $/฿	20,78 Mio.
Sheepmarket	Darknetmarktplatz	2013	5400	BTC	1047,90 $/฿	5,66 Mio.
PicoStocks	Börse	2013	5896	BTC	510,41 $/฿	3,01 Mio.
Inputs.so	Webwallet	2013	4000	BTC	195,84 $/฿	783.360
BIPS	Webwallet	2013	1295	BTC	510,39 $/฿	660.959
Bitcash	Coinbörse	2013	484	BTC	511,20 $/฿	247.422
Vicurex	Coinbörse	2013	1454	BTC	112,35 $/฿	163.351
Ozcoin	Miningpool	2013	922	BTC	114,53 $/฿	105.600
BTCGuild	Miningpool	2013	1254	BTC	57,56 $/฿	72.556
BitLC	Miningpool	2013	2000	BTC	25,74 $/฿	51480
Bitstamp	Coinbörse	2014	18866	BTC	279,00 $/฿	5,26 Mio.
SilkRoad2	Darknetmarktplatz (Betrugsverdacht)	2014	4400	BTC	616,05 $/฿	2,71 Mio.
BTER (1)	Coinbörse	2014	51,67 Mio.	NXT	0,03 $/NXT	1,65 Mio.
FlexCoin	Webwallet	2014	896	BTC	823,93 $/฿	738.240
Purse.io	Shopping	2015	10235	BTC	248,91 $/฿	2,55 Mio.
BTER (2)	Coinbörse	2015	7170	BTC	244,07 $/฿	1,75 Mio.
Coinapult	Webwallet	2015	150	BTC	286,67 $/฿	43.000
Cavirtex	Coinbörse	2015	-	BTC	- $/฿	-
Bitfinex	Coinbörse	2016	119756	BTC	593,70 $/฿	71,10 Mio.
The DAO	Smart Contract	2016	360000	ETH	17,32 $/ETH	62,35 Mio.
Gatecoin (ETH)	Coinbörse	2016	185000	ETH	10,63 $/ETH	1,97 Mio.
Gatecoin (BTC)	Coinbörse	2016	250	BTC	457,41 $/฿	114.352
Shapeshift	CoinWechselstube	2016	-	Misc.	-	230.000
Cointrader	Coinbörse	2016	-	BTC	- $/฿	-
Bitthumb	Coinbörse	2017	-	Misc.	-	82,7 Mio.
Tether	ICO	2017	30,95 Mio.	USDT	1,00 $/USDT	30,95 Mio.
Parity Wallet	Ethereum Client	2017	150000	ETH	200,00 $/ETH	3,00 Mio.
CoinDash	ICO	2017	43000	ETH	176,74 $/ETH	7,60 Mio.
Veritaseum	ICO (Betrugsverdacht)	2017	36000	VERI	205,49 $/VERI	7,40 Mio.
Enigma.co	ICO	2017	1500	ETH	300,00 $/ETH	450.000
Coincheck	Coinbörse	2018	523 Mio.	NEM	1,02 $/NEM	534 Mio.

doi: 10.2313/NET-2018-03-1_06

B. LISTE VON UNTERSUCHTEN BETRUGSFÄLLEN

Fall	Kategorie	Jahr	Anzahl Coins	Cointyp	Damaliger Wechselkurs	Höhe $ damals
Bitcoin Savings & Trust	HYIP	2011	146000	BTC	5,53 $/฿	807.380
Ubitex	Startup	2011	1139	BTC	13,62 $/฿	15.515
MyBitcoin	Coinbörse	2011	78740	BTC	13,62 $/฿	1,10 Mio.
Bitscalper	HYIP	2012	1350	BTC	4,73 $/฿	6461
GBL	Scam	2013	22000	BTC	195,95 $/฿	4,31 Mio.
MyEtherWallet	Webwallet	2017	-	ETH	-	-
Richmond Berks	HYIP	2017	-	BTC	-	-
Ethtrade	HYIP	2017	-	ETH	-	-
Neo BTC Bank	HYIP	2017	-	BTC	-	-
Nordebank	HYIP	2017	-	BTC	-	-
7thirty	HYIP	2017	-	BTC	-	-
Iotaseed	Phishing/Fake App	2018	-	IOTA	-	-

doi: 10.2313/NET-2018-03-1_06

WeSee: dynamic visualization of Web Service use

Sergey Podanev
Advisor: Miguel L. Pardal
Seminar Future Internet WS2017/2018
Chair of Network Architectures and Services
Departments of Informatics, Technical University of Munich
Email: sergey.podanev@tum.de

ABSTRACT

The software applications that we use in our daily lives are typically not isolated when running in our phone or computer. In fact, they may connect to remote resources through REST or SOAP Web Services. Many times, programmers are not aware of this network activity, as the activity is done by libraries used by the application. The end-users are even more unaware of this reality. Usually these remote invocations are benign, but there is still a problem of transparency. To address this concern, we present *WeSee*, a tool designed to intercept network connections, display them in a graphical manner and, through this data visualization, make visible what is hidden. The goal is to raise the awareness of developers and end-users about the network interactions of the applications that they use.

This paper describes the technical details of the *WeSee* tool implementation, available as open-source [1]. The prototype is functional and can be used to capture and display network traffic in a single device. The architecture is extensible and, in the future, the system can be used for intercepting different types of network payloads and for monitoring multiple devices at the same time.

Keywords

Data visualization, Network Monitoring, Packet Interception, Privacy-awareness.

1. INTRODUCTION

Today, we cannot imagine our life without using modern information technologies. In the origins of computer science, programs ran on a single device and were relatively simple, as the machines did not have much computing resources and the code base was moderate in size. All variables were traceable and the program behavior could be verified by the programmer.

Since those early days, software has grown significantly in size, and in complexity [2] [3]. Modern programs use many layers of code, and programmers are divided into special fields. The full program algorithm is no longer transparent, even for experienced programmers. Debuggers and other tools can be used to follow the flow of execution, but this task is too complex and time-consuming.

Programs are composed of many libraries that are combined to produce the desired functionality. The use of third-party libraries is essential to save time and benefit from external expertise. But such use poses risks [4].

The issue is not just the use of libraries. These libraries can access the Internet and communicate with remote servers. Programmers are potentially unaware of the data being transmitted. Consider the following example: to display a map in the application, the developer is using a well-known maps library. To retrieve a map, location coordinates are shared with the service. Additionally, other data can be sent to other remote servers. For the developer, conceptually, the program is "just" showing a map, but, in reality, it is also sending usage data to a third party. This is a significant privacy concern.

1.1 Application monitoring

The first approach to monitor applications is to monitor network connections and select suspicious ones. There are "sniffing" tools such as *Network Inventory* and *TcpDump* in the operating system. *Network inventory* is a tool, with a GUI interface used to collect all network data from connected devices and get all operating system and device statistics [10]. *TcpDump* is a TCP/IP packet analyzer that runs from the command line [11]. *Wireshark* can be a good alternative for monitoring connections. It shows standardized values of packet fields with particular values (given in text and byte form). However, *Wireshark* cannot be adapted for other graphical representations (e.g. graphs) and cannot be directly used for "sniffing" particular parts of the code [9].

The other possible solution of handling suspicious code is to "jail" applications. There are several projects which isolate applications, for example *Firejail* [6] or *Linux containers* [7]. In addition, firewalls [8] can also be used to contain network traffic. Unfortunately, people cannot manually supervise all of connections, therefore it is difficult to manually set all necessary filters into the firewall without proper analysis. All applications can be "jailed". But, in this case, the restriction policy for each of the applications should be applied accurately. Sometimes, it is required to create connections with new destinations, but these new destinations might have been banned by a user previously. The content of the connections can be encrypted and not properly described by the general solution. The observed tools have a general approach for each technology and do not allow centralized monitoring of a network at different OSI model[1] levels from different devices.

[1]Open system Interconnection

Seminars FI / IITM WS 17/18,
Network Architectures and Services, March 2018
45
doi: 10.2313/NET-2018-03-1_07

Table 1: Requirements list

#	Requirement
R1	Capture network payloads
R2	Store the payloads persistently
R3	Display the payloads in a graph inspired notation, where nodes are devices, and edges are messages between devices
R4	Allow the filtering and selection of data to display
R5	Allow inspection of captured message details
R6	Allow diverse types of payloads to be captured, at different levels in the OSI model (link, network, transport, application)
R7	Provide a user interface aimed primarily at software developers, but that can also be used by end-users.

If any of these tools lead to suspicion that something is wrong, then it is time to reconfigure and use firewalls and other security tools, such as an anti-virus, malware detectors, intrusion detection systems, etc.

1.2 Network visualization

The main idea of our approach is to use *visualization* of data from passive monitoring of selected parts of the network, to capture messages, and to present these messages in a way that can be understood by humans. The goal is to show network activity and, through this means, to raise the awareness of the users about the communication dependencies of the applications.

In this paper we present *WeSee*, a tool designed to meet the requirements stated in Table 1. The system designed is presented in detail in the next section.

2. SOLUTION

Intercepting communications with a universal interface allows collecting useful data from different sources with central node. Based on this data, statistics of network nodes and connections can be calculated. And finally, a graph is used to illustrate processed information. The listed functionality was reached by designing the next general deployment structure.

The *WeSee* tool performs the following next tasks:

- Intercept communications with a universal interface;
- Gather and calculate overall simple statistics;
- Display connections with a coherent graph.

2.1 Design

The design of *WeSee* is shown in Figure 1.

The `intercepted device` is concerned with the data interception and capture near the target application. To intercept new connections, an `interceptor` module is integrated inside the application to log it.

The role of the `Visualization server` is to collect and process data, collected from interceptors. It is implemented as

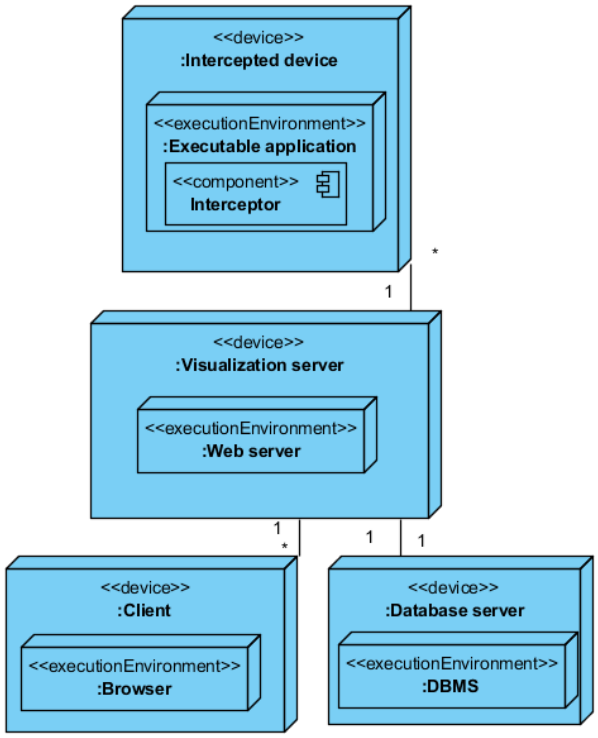


Figure 1: WeSee deployment diagram

a web server. A web server is most suitable for that purpose as it can serve multiple clients and provide a universal graphical interface as a web page, which is accessible from most modern devices.

The `database server` helps the web server to collect and process data with concurrent access.

A `browser` can be used to visualize the data, as it is supported by almost all computer platforms. Thereby, a user can monitor remotely and from multiple devices. When a user loads a web page, the browser downloads all requested data from the web and then draws a suitable graph.

The general algorithm is illustrated in Figure 2. Data is stored into a database and can be later collected, filtered and converted for presentation with statistics.

2.2 Implementation

The implementation of the *WeSee* tool consisted of several steps. At first, it was necessary to select suitable technologies to implement it, define program functional possibilities and therefore to narrow possible architecture solutions. As the tool consists of the several deployment parts, on the second step it is important to define the general architecture to connect these parts into the tool. Lastly, every program module can be implemented separately according to the general architecture.

2.2.1 Selected technologies

WeSee was implemented in Java, a high level programming language, which speeds up development, is platform inde-

Seminars FI / IITM WS 17/18,
Network Architectures and Services, March 2018
46
doi: 10.2313/NET-2018-03-1_07

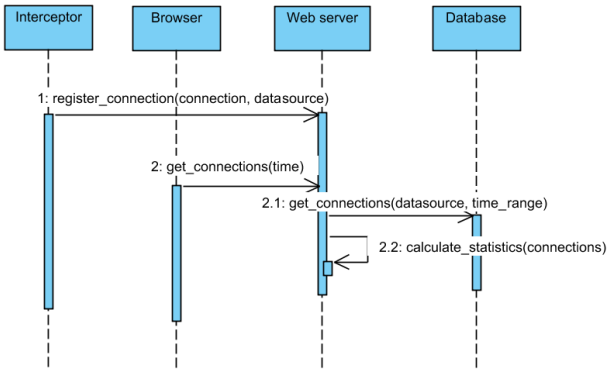


Figure 2: WeSee general sequence diagram

pendent, reliable, sufficiently optimized for the project task and maintainable due to its high popularity and longtime presence [12].

The connection interception is done by each monitored application individually. For instance, it can be implemented directly into the code of the target application.

To send the data, REST services are used. REST services use uniform stateless interfaces to register new connections [13]. Also, the REST service is used for requesting all occurrences for the selected connection in a given time range.

The alternative is SOAP services, however, currently SOAP has a downward popularity trend, because it is more complex to configurate [14].

To develop the web server, the *Spring Boot* framework was selected. It is a framework that provides much functionality, including REST services, and database connections support [15]. To connect to the database, *Hibernate* technology was used. *Hibernate* maps database tables to Java objects automatically, simplifing the database usage [16]. The underlying database was *MySQL*. It is a database with a long history, which obtained a rich support of connection technologies, including *Hibernate*. It provides a multiple thread access mode, has sufficient performance and is supplied with a community edition. As already mentioned the database can be replaced with any other as the result of using the *Hibernate* technology [17].

To represent the data in a visual way, the *D3* library was used [18]. *D3* (data-driven documents) has a rich interface for creating graphs on the SVG HTML element. This library provides powerful components to create dynamic graphs with a force simulation, which makes the visualization interactive. To draw the network graph, a force-directed graph is included in the *D3* library. A force-directed graph is the most common way to draw a network topology. It is a weighted graph with sets of vertices connected by edges. The edges as well as vertices can have their own number(weight). These weights are used to illustrate the number of messages that were transmitted over a connection or a network node.

This idea to use this graph was inspired by the Gource project, where an animation of contributions to a Git repository can be generated [20]. The other good example is vizceral project [21]. It builds an orientated graph of network traffic volume.

2.2.2 *Persistence*

The data layer is shown in Figure 3.

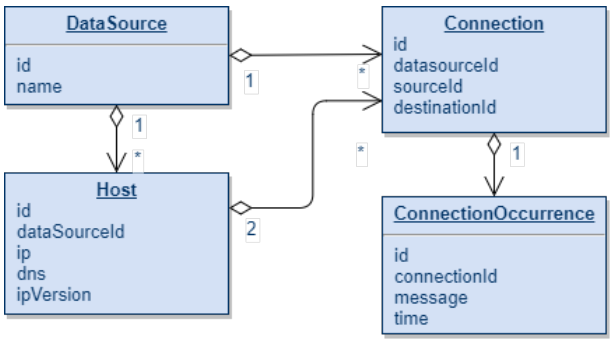


Figure 3: Database schema (relational model)

The database stores the general information about occurred connections. But connections can be received from different interception sources and at the same time they are described for the same network hosts. To represent connections on the graph, the collected information should be merged from the selection of the multiple data sources.

2.2.3 *Interception*

In order to intercept the data, a universal interface should be provided for each intercepted application. This interface sends the data on a REST service, which can then store the data on the server. This interface is represented in Figure 4. It contains an abstract interface with its concrete implementation and a set of data definitions to describe intercepted connections.

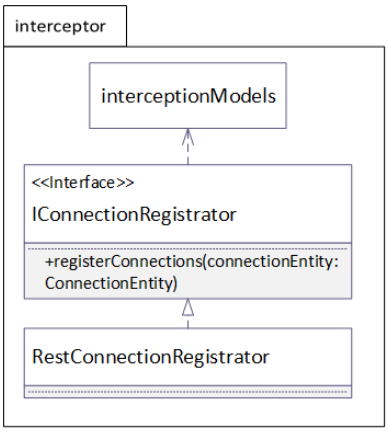


Figure 4: Interception module structure

Seminars FI / IITM WS 17/18,
Network Architectures and Services, March 2018
47
doi: 10.2313/NET-2018-03-1_07

2.2.4 *Pcap interception*

In order to have a rich functional test and present a tool use case, a `pcapInterceptor` was developed as implementation of the `interception` module.

To intercept a connection, it uses the *pcap* library. *Pcap* is distributed for multiple operation systems platforms including Windows and Linux under different names [19]. The library "sniffs" all selected interfaces and finds network packets with a configured filter.

We had to decide at which level to capture network activity. Different levels have different advantages and disadvantages. The lower network levels provide more information, but usually there is more encoded data, which is not easily understandable. The higher network levels have more concise data but do not have identification of hosts and other details. The network layer contains necessary information for creating a network graph, which includes especially IP addresses, and even information from higher levels, for example, of the transport or application layer. Therefore, network layer is the most suitable for recording connection metadata. If the higher network packets can be extracted by the *pcap* library, then `pcapInterceptor` the packet payload from these levels is recorded as a sent message. As a result, the pcap library provides a rich data set for visualization even for one computer. The structure of the `pcapInterceptor` application is illustrated in Figure 5.

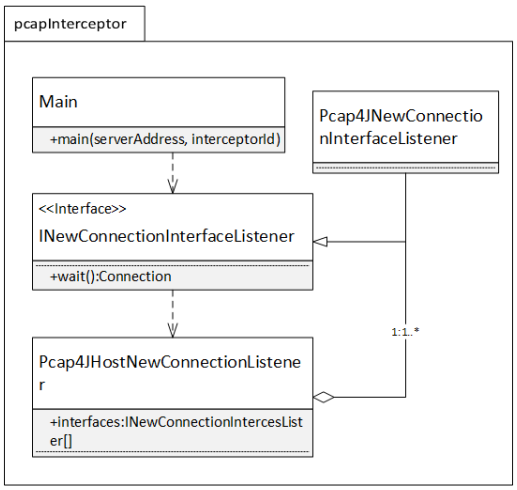


Figure 5: PcapInterceptor application class diagram

The main function is used to start the application, which repeatedly catches packets from sniffed interfaces in different threads. The `Pcap4JNewConnectionInterfaceListener` interface allows to refer one to a computer listener as a united object of combination of individual interface listeners. In case the volume of the packets is too large to report to the server, they are skipped. As the application works in an infinite loop, it refreshes interfaces periodically as shown in Figure 6.

2.2.5 *Visualization server*

The `visualization server` is the central node of the tool as it implements the functionality of gathering, analyzing and representing the data. Similarly to the logical division,

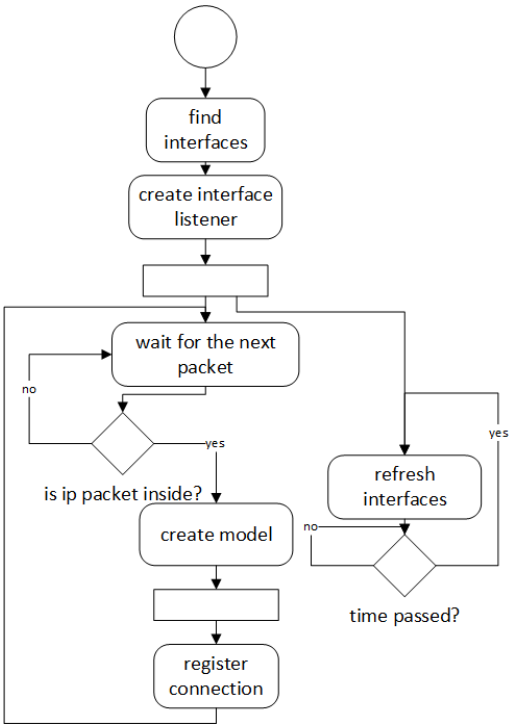


Figure 6: PcapInterceptor behaviour

WeSee data definitions were separated into a layered architecture according to Figure 7).

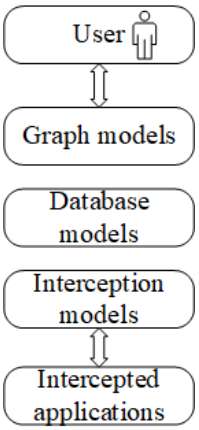


Figure 7: Visualization server layering

Intercepted applications send the data in a JSON format, which is different from the format stored in the database. Both data structures contain information about the hosts participating in the communication and the message payload. Applications which send the JSON data do not excessively put with information about themselves and send plain information about connections. In the database all received data is stored with data sources (application identifications) and indexed.

Graph layer data structures are a representation of the statistics, since the information about nodes from different data

Seminars FI / IITM WS 17/18,
Network Architectures and Services, March 2018
48
doi: 10.2313/NET-2018-03-1_07

sources should be merged to present the composite picture of the network. The server back-end consists of 3 modules: `srv.rest`, `graph` and `repository`, depicted in Figure 8.

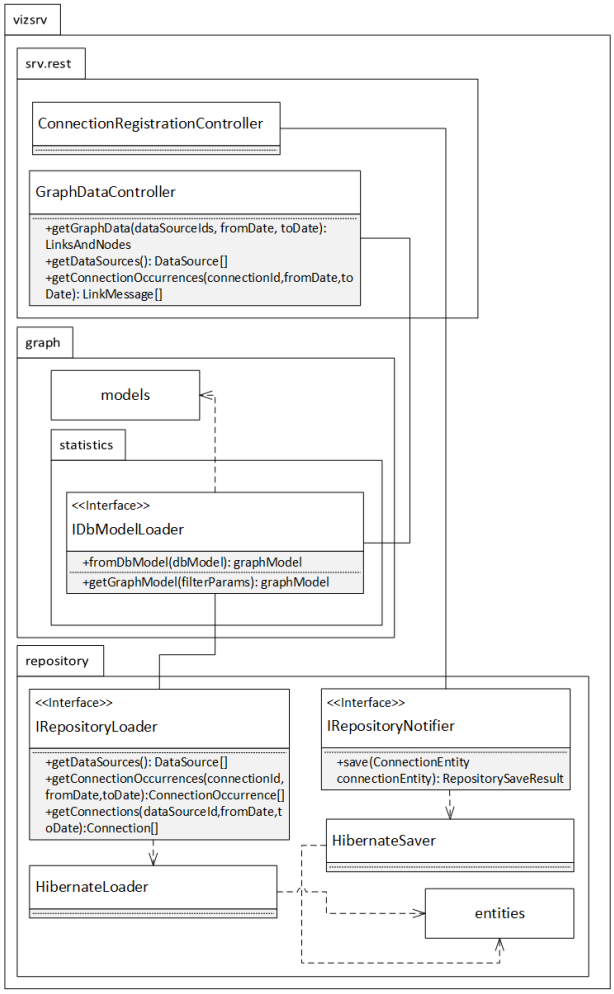


Figure 8: Visualization server class diagram

The `srv.rest` module consists of Spring REST service implementations. They interact with a browser to provide the data with asynchronous requests (for example, in case of filtering by the date/time and data sources) and collect the data from intercepted applications.

`Graph` and `repository` modules work with the graph and database data definitions respectively and convert the data to the proper form for controllers. The `database` module is based on the *Hibernate* technology. The application database connector enables to use *Hibernate* and contains configuration to connect the database. In addition to the connector, the database with a user account was created. That database user is delegated to the `visualization server` application. The information about database is placed in the *Hibernate* configuration file of the `visualization server`.

2.2.6 *Presentation*

To present the data, HTML was used. When the page is loaded the client makes requests to the server. Generally, the web page plays the role of a single-page application due to the narrow specialization use.

To configure a node spacing on the graph several forces were created to push unconnected nodes or magnify nodes keeping the link distance. The width of the links and the radius of the nodes scales with the power function. This limits the maximum size, but shows an object which can be scaled, depending on the object activity. The client-side application is written in JS (JavaScript) and consist of several parts represented in Figure 9.

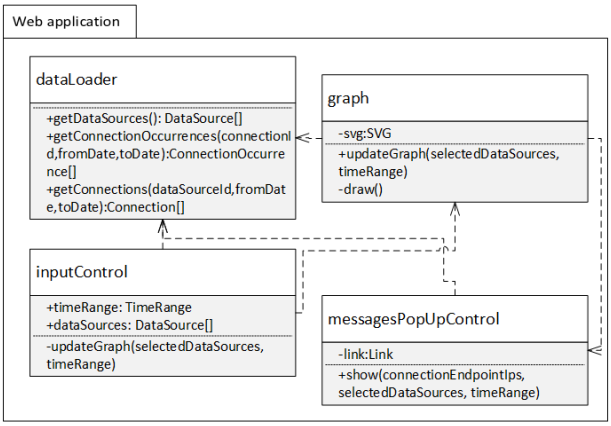


Figure 9: Web application structure main objects

The `dataLoader` is a JS object and is responsible for downloading data from the web server with requests to the server. Downloaded data can then be filtered with local search parameters and reorganized to fulfill the D3 library data structure requirement. The D3 library defines the special format of nodes and links for the force-directed graph. Graph control uses the SVG HTML element to draw vector graphic elements. The `InputControl` object generates possible filtering ranges and acts on entering new filter parameters for the data selection. It controls the limits of the possible data range for selected data sources and refreshes the graph on update. `MessagesPopUpControl` shows a pop up window with detailed connection occurrences. The user can list particular messages transmitted over a given time from selected data sources. To provide an opportunity of filtering and ordering data, the datatable JS library was used [22]. It allows a user to search by a plain string search or reorder rows by the columns: time, data source or message text.

3. WESEE IN ACTION

This section presents an example of the *WeSee* tool using the `pcapInterceptor`.

Figure 10 shows the start page. A user is able to examine the start page with the full selected data range by default. Then, it is possible to narrow the search results by selecting the names of data sources or pick the time range. The graph consist of nodes that represent network interfaces of the devices; and links show connections between them. On hovering, a pop-over message shows brief information about the host: IP, usage statistics, DNS and link information such as the number of messages host sent and received, last message, end-point addresses. The next steps show one way of

Seminars FI / IITM WS 17/18,
Network Architectures and Services, March 2018
49
doi: 10.2313/NET-2018-03-1_07

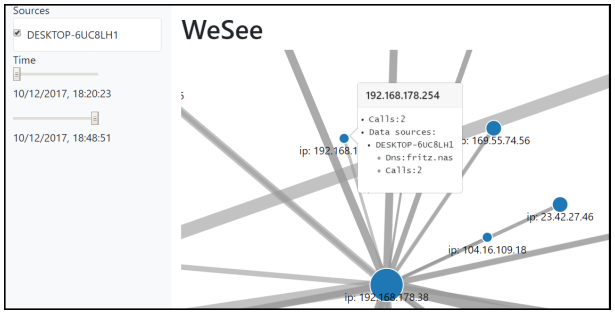


Figure 10: WeSee network graph screenshot

using the tool and demonstrates the functionality, stated in the requirements.

With the selected time range from the source "DESKTOP-6UC8LH1" it was discovered that a computer sent two messages to the DNS "fritz.nas", which actually is the name of a router (Figure 10). This information can be used for node identification and for exploring its activity.

On double clicking the link, a pop up window appears where all intercepted messages are listed with information of the date/time occurrence, name of interceptor, brief and full text of the sent message, as shown in Figure 11. It was

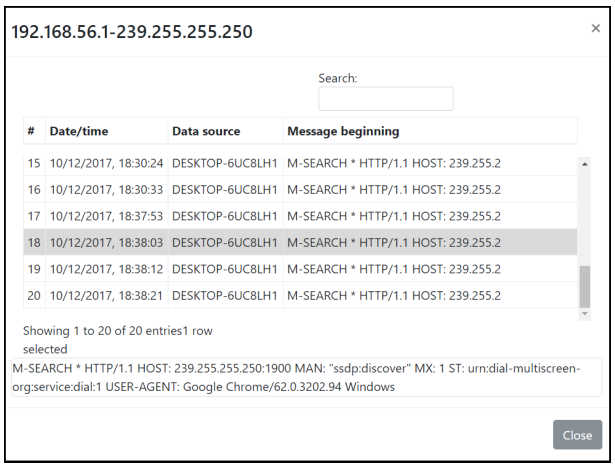


Figure 11: Detailed information about connection

discovered, that there is a link to the multicast IP address 239.255.255.250 and it primarily contains SSDP (Simple Service Discovery Protocol) messages. SSDP is used for searching UPnP devices in the home network [23]. The Google Chrome browser performed this action automatically after opening. Although this connection was initiated by an authorized application with a standardized protocol, the *WeSee* tool also can detect other hidden connections and depict it on the graph. Observing this graph, the awareness is raised, as it shows the entire network with elementary statistics, to which the computer is connected and sent messages within the given period of time.

Table 2: Requirements assessment.

#	✓/x	Comments
R1	✓	Network payloads are captured in `pcapInterceptor` for network level and higher.
R2	✓	Information is stored in a database
R3	✓	Implemented orientated graph with D3 library with nodes as hosts and edges as connections.
R4	✓	Filtering by data sources and time range. For selected connection: reordering messages, search by plain text.
R5	✓	Provided message details: occurrence time, data source, short and full message text.
R6	✓	`pcapInterceptor` captures payloads of network level and higher. The API allows to collect information of any layer, but should be converted to the defined data structures.
R7	✓	Provides user interface as a web page.

4. CONCLUSION

After creating a working prototype of the *WeSee* tool, it is necessary to check fulfilled requirements from the Table 1. As it can be seen from the Table 2, all requirements were reached as discussed in the comments column.

The *WeSee* example demonstrated, that even a single device can participate in a large number of network connections. As more interceptor implementations become available, for different network levels and protocols, and for different devices, the tool has potential to make visible even more interesting pictures of application activity.

The project is open-source and can be used by any programmer, who is concerned about network security and wants to reveal the "secret life" of his/her own devices [1].

In the author's opinion, with the trend of increasing heterogeneity in programming environment, *WeSee* will only become more relevant.

4.1 Future deployments

The following scenarios show how *WeSee* can be used in different deployments, to provide visibility of network activity.

4.1.1 SOAP web service

This is a specific example of a different kind of payload, in this case, at the application level. SOAP messages can be used for web services [14]. However, it is also necessary to track SOAP communication, as the information can be sent to the unknown web server or content of the message can be suspicious. The SOAP messages are more high-level then the packets, and, as a result, are easier to analyze. To integrate, a SOAP message listener can connect to the "interceptor" module, fill "interceptor" data structures and send collected data with this interface. It is also possible to create a specialized interceptor, that can be connected by adding a few lines to the standardized server configuration file.

In this deployment also, multiple service invocations can be captured and sent to the same visualization server, to pro-

Seminars FI / IITM WS 17/18,
Network Architectures and Services, March 2018
50
doi: 10.2313/NET-2018-03-1_07

vide an overview of the interactions between a set of user services.

4.1.2 IoT gateway

The lack of visibility of network activity can be further aggravated in the Internet of Things (IoT). These devices show a great potential with their ability to exchange information about environment and use it for interaction [26].

In the IoT, the programs are running on devices not usually considered as computers, like light bulbs or door bells. There is also no visibility on the network activities of all these devices. They can share their information not only with an authorized server, but with someone else too. Although, mostly these information is not security critical (for example, the temperature in home), but can indirectly harm too.

IoT software can be modified, but not if it is embedded in devices. The approach is redirecting the traffic from the used gateway. Typically, private networks, which usually used for IoT, have gateways to the external network. Modern routers support a traffic redirection feature. After that, a redirected traffic can be analyzed by an interceptor application on other computer and the result can be collected by a central *WeSee* server. However, this method can be problematically deployed for the network with many gateways. In this instance, it is required to configure multiple gateways and keep their configuration up-to-date.

4.2 Future improvements

The current state of the tool fulfills its main requirements. However, the project can be extended to improve the user experience and provide more functionality. Currently, the content of network packets is not decrypted, but unencrypted information is avaliable. It is done due to the wide specialization of the current interceptor in order to demonstrate tool possibilities. It can be fixed by implementing a decrypting module on the interceptor side or including build-in decryption into the server.

The other way is to use specialized interceptors, for example, which intercept REST service messages for the selected path.

Additionally, a node's geolocation identification can be added into the tool. The geolocation identification potentially raises the user's awareness about a host location as an object connected with "real" world places, not abstract IP addresses.

With the inclusion of extra search parameters, the web application can be built using one of the single-page frameworks, like AngularJs or OpenUI5 [24][25]. The *WebSee* web application already works as a single-page application. Currently, the amount of JS code is small, but on future releases the structure can be better organized with use of existing reliable frameworks. They can improve responsiveness and maintainability of the front-end side of the application.

We hope that, in the future, more use of *WeSee* is made, and that a *gallery* of interceptors and visualizations can be built.

WeSee also has potential as a learning tool, for example, as part of advanced and interactive teaching methods [27].

5. REFERENCES

[1] *Github repository: WeSee: dynamic visualization of Web Service use, last access: 21.01.2018*, `https://github.com/inesc-id/WeSee`

[2] Amit Deshpande, Dirk Riehle: *The Total Growth of Open Source*, Open Source Development, Communities and Quality. OSS 2008. IFIP – International Federation for Information Processing, vol 275. Springer, Boston, MA, 2008

[3] Barry Boehm: *A View of 20th and 21st Century Software Engineering*, Proceedings of the 28th International Conference on Software Engineering, Shanghai, China, May 20-28, 2006

[4] Steven Raemaekers, Arie van Deursen, Joost Visser: *Exploring Risks in the Usage of Third-Party Libraries*, Software Improvement Group, Delft University of Technology, Amsterdam, The Netherlands, 2015

[5] *Global Internet Report 2016, Internet society, pages 31-34, 2016, last access: 21.01.2018*, `https://www.internetsociety.org/globalinternetreport/2016/wp-content/uploads/2016/11/ISOC_GIR_2016-v1.pdf`

[6] *Firejail security sandbox, last access: 21.01.2018*, `https://firejail.wordpress.com`

[7] *Linux containers, last access: 21.01.2018*, `https://linuxcontainers.org`

[8] *Internet Firewalls: Frequently Asked Questions, last access: 21.01.2018*, `http://www.faqs.org/faqs/firewalls-faq/`

[9] *Wireshark, last access: 21.01.2018*, `https://www.wireshark.org`

[10] *Network inventory advisor, last access: 21.01.2018*, `https://www.network-inventory-advisor.com`

[11] *TCP dump, last access: 21.01.2018*, `https://www.tcpdump.org/`

[12] *Java compared with other languages, last access: 21.01.2018*, `https://www.safaribooksonline.com/library/view/learning-java-4th/9781449372477/ch01s03.html`

[13] *REST (representational state transfer), last access: 21.01.2018*, `http://searchmicroservices.techtarget.com/definition/REST-representational-state-transfer`

[14] Pavan Kumar: *On the Design of Web Services: SOAP vs REST*. UNF Theses and Dissertations.138, pages 58-61, University of North Florida, USA, 2011

[15] *Spring framework, last access: 21.01.2018*, `https://spring.io`

[16] *Hibernate, last access: 21.01.2018*, `http://hibernate.org`

[17] *MySql open source database, last access: 21.01.2018*, `https://www.mysql.com`

[18] *Data-driven documents library, last access: 21.01.2018*, `https://d3js.org`

[19] *Pcap - Java library for capturing, crafting, and sending packets, last access: 21.01.2018*, `https://www.pcap4j.org`

[20] *Gource, software version control visualization, last access: 21.01.2018*, `http://gource.io`

doi: 10.2313/NET-2018-03-1_07

[21] *Vizceral, WebGL visualization for displaying animated traffic graphs, last access: 2.02.2018*, `https://github.com/Netflix/vizceral`

[22] *DataTables, Table plug-in for jQuery, last access: 2.02.2018*, `https://datatables.net/`

[23] *Simple Service Discovery Protocol/1.0, last access: 21.01.2018*, `https://tools.ietf.org/html/draft-cai-ssdp-v1-03`

[24] *Angular js framework, last access: 21.01.2018*, `https://angular.io`

[25] *OpenUI5 - open source js UI library, last access: 21.01.2018*, `http://openui5.org/`

[26] Dieter Uckelmann, Mark Harrison, Florian Michahelles Florian: *An architectural approach towards the future Internet of Things*, Architecting the Internet of Things, pages 1-24, Springer, 2011|

[27] Marc-Oliver Pahl: *The iLab Concept: Making Teaching Better, at Scale*, IEEE Communications Magazine, vol. 55, no. 11, pp. 178-185, November 2017

doi: 10.2313/NET-2018-03-1_07

Hybridization of Neural Networks and Genetic Algorithms

Ethan Tatlow
Advisor: Márton Kajó
Seminar Future Internet WS 2017/18
Chair of Network Architectures and Services
Departments of Informatics, Technical University of Munich
Email: tatlow@in.tum.de

ABSTRACT

Artificial neural networks are a versatile type of computing system that are capable of learning to compute functions through observation, as opposed to explicitly declaring how the function is meant to be calculated. When designing such artificial neural networks, a great deal of effort goes into choosing the parameters of such a network, with much trial and error often involved due to the ever increasing complexity of modern networks. This paper presents how genetic algorithms, a class of biologically inspired optimisation algorithms, can be used in combination with neural networks to help with such choices, as well as the effectiveness and the limitations of some of these combinations.

Keywords

Neural Networks, Genetic Algorithms, Hyperparameter Optimisation, Neuroevolution

1. INTRODUCTION

In the field of artificial intelligence, many concepts repeatedly go through popular and less popular periods. Neural networks are a prime example of this: based upon a reasonably simple basic concept that has in the meantime existed for over 70 years, they are once again receiving a heightened amount of attention from the current researching commmunity. This is, amongst other reasons, due to the success of so called deep neural networks, which are typically highly complex neural networks capable of solving equally complex problems [11]. Because of this complexity, ever more effort has to go into configuring these systems for them to perform well. This is a problem, as the exact effect of modifications to certain configuration parameters to the system's performance are often unknown, requiring much manual trial and error before any sort of success can be declared [11].

One way of solving this problem is to automate the choice of such parameters. Using genetic algorithms, a subclass of the optimisation algorithm class of evolutionary algorithms, is one method for which this can be done in a reasonably efficient manner, at least for some of the parameters [11, 8]. In the field of neuroevolution, networks are trained using genetic algorithms, as opposed to the more traditional method of gradient descent using backpropagation, while some approaches additionally modify the structure during the training process [13]. In the following, we shall introduce how some such applications of genetic algorithms to neural networks can be made, with the goal of giving the reader a generic idea of the usefulness of such methods and of their limitations.

2. NEURAL NETWORKS

As the name suggests, artificial neural networks (hereafter referred to simply as neural networks) are based upon a model of the large networks of neurons found in mammalian brains (an adult human brain, for example, has approximately 86 billion neurons [1]). The fundamental idea was first introduced in a paper by neurophysiologist Warren McCulloch and Mathematician Walter Pitts written in 1943 on how neurons might work, modeling a simple neural network with electrical circuits.

The main advantage of neural networks is that, much like a human brain, they are able to adapt and learn with given training data. Neural networks possess limitations, of course; although popular media often depicts them differently, neural networks are neither capable of thinking in the same way human brains can, nor are they some miracle of computing that is capable of solving all of the world's hardest computational problems [3]. However, their adaptive learning properties make them very useful for solving many types of problems for which an algorithm, or at least an efficient algorithm, doesn't exist or can't be found, but sufficiently large sets of data are present with which a network can be trained.

The author would like to note at this point that there is much to be found on the topic of neural networks, and that attempting to introduce all variations and their implementations in detail would go beyond the scope of this paper's topic. Consequently, this chapter shall attempt to give a basic introduction by focusing on the most commonly encountered variant, namely feedforward multilayer perceptrons.

2.1 Learning Process

One of the great advantages of neural networks is their ability to learn; in the field of machine learning, two different kinds of learning are generally distinguished: supervised and unsupervised learning [4].

Supervised learning is achieved with a set of training data consisting of pairs (x, d_x), with x being an input to a neural network, and d_x being the the desired output for the given input x. The actual output y of the network is then compared to the desired output d_x, and, based upon that, the network's parameters are adjusted.

doi: 10.2313/NET-2018-03-1_08

Unsupervised learning works differently: the training data lacks the desired results (also known as labels) d_x. Instead of learning to give a specific output for a given input, the network instead forms internal representations of the input data, encoding certain features of the input based upon the learning rule used by the network.

2.2 Application Example Digit Classification

In order to give an idea of what kind of problems are well suited to having neural networks applied to them, we shall consider the problem of digit classification: presented with a series of handwritten digits between 0 and 9, a human will usually have no problem recognizing them, more or less independent of the actual handwriting. When attempting to conceive an algorithm to perform this task, it doesn't take long to realise that it is hard to transform simple concepts that humans use such as "a 1 is more or less shaped like a line", "a 0 has an ellipse shape" or "a 9 looks like a circle with a squiggly bit at the bottom" into code. Assuming one has a 16 pixel by 16 pixel image, one would have to consider 256 pixels individually, their relationship to one another, allow for tolerances, and at the end still have a result that is vastly complicated to debug and may well only work properly for certain sets of handwriting that are considered in the algorithm's implication.

When presented with the same problem, a simple neural network requiring minimal implementation effort, on the other hand, can, given enough training data, recognize digits with an accuracy of around 95% [2], almost equally independent from handwriting as a human (assuming sufficient different sets of handwriting are present in the training data).

2.3 Artificial Neurons

Neural networks are composed of artificial neurons, with the number and their exact structure in a network varying depending on application and implementation. There are several different types of neurons that can be used, the simplest type being the perceptron. This artificial neuron was developed by Frank Rosenblatt in the 1950s and 1960s, and is based upon the artificial neuron model introduced by McCulloch and Pitts in 1943[4].

For the neurons we shall be considering, a neuron can be described as having a bias $b \in \mathbb{Z}$, $n \in \mathbb{N}$ binary inputs x_1, x_2, ..., x_n, each with a corresponding weight $w_i \in \mathbb{Z}$, and producing an output y. The output is determined through summation of the inputs and the bias, and passing this sum through an activation function ϕ. For perceptrons, the output y is thus determined in the following manner:

$$y = \begin{cases} 1, & \text{if } b + \sum_{i=1}^{n} w_i x_i > 0 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

An interepretation of this is that the weights of the individual inputs indicate their relevance to this neuron's output, while the bias indicates the neuron's tendency to 'fire': given the same set of weights, a less negative bias will result in a higher likelihood to give an output of 1 than a more negative bias.

Although this neuron is simple and reasonably straightforward to understand, it has disadvantages making it unfavourable for usage in many applications [2], one of those being the lack of continuity in it's output. We shall therefore introduce a second type of neuron, the sigmoid neuron. This neuron gets it's name from the sigmoid function σ used as an activation function:

$$\sigma(z) = \frac{1}{1 + e^{-cz}} \tag{2}$$

with c being a fixed constant. For a sigmoid neuron, the output signal y is now computed in the following way:

$$y = \sigma(b + \sum_{i=1}^{n} w_i \times x_i) \tag{3}$$

While at a first glance this neuron doesn't seem to have too much in common with the perceptron, a closer look at the sigmoid function shows $\lim_{z \to -\infty} \sigma(z) = 0$ and $\lim_{z \to \infty} \sigma(z) = 1$, with the function having a distinctive s-like shape (hence the name). The effect of this is that instead of simply outputting either 0 or 1, a sigmoid neuron outputs values close to 0 for large negative values for z, and values close to 1 for large positive values for z; for such values for z, it mirrors the output of a perceptron, while still allowing for intermediate values. Put differently: while perceptrons are only capable of outputting 'yes' and 'no', sigmoids are capable of outputting different degrees of 'maybe', or instead of only differentiating between, say, a black and white pixel, differentiating between different degrees of grey. This may not always be ideal; in the previously used scenario, the waiter won't typically want to hear the phrase 'I think I might have the steak with a tendency of 0.7'. In such a case, however, it is reasonably simple to decide on an appropriate rule for interpreting the output along the lines of 'anything above 0.5 shall be interpreted as a 1'.

2.4 Network Topology

So far, we have only considered individual neurons, which on their own can model a simple decision making process. We are, however, dealing with entire networks of neurons, not just individual neurons; this means that the inputs of a neuron and, consequently, it's output, depend on the outputs of other neurons, which in turn also may depend on the outputs of still other neurons and so on. This allows for a higher complexity in decision making, giving the possibility to model more complex associations between the inputs and outputs of a network, with neurons further from the 'original' inputs being capable of making more complex and decisions through abstract internal representation of information [2].

The networks that we will consider will contain two restrictions:

- the neurons shall be arranged in layers, with each layer's outputs only being used as inputs for neurons in the next layer

doi: 10.2313/NET-2018-03-1_08

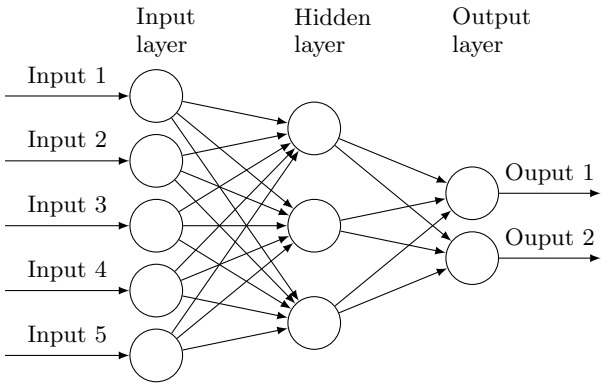


Figure 1: An example neural network architecture

- all neurons in a layer shall receive all of the prior layer's neurons' outputs as inputs

This means, among other things, that there are no connections between neurons within the same layer, and that there are no cycles in our network, making the network a so-called feedforward network; this means that information is only ever passed forward through the network [2]. An example network with these properties can be seen in figure 1.

In this depiction, a circle represents a neuron, while an arrow going from one neuron A to another neuron B is meant to indicate that the output from A is used as an input for neuron B; note that the neurons in the first layer, referred to usually as the input layer, only have one input each, and that these inputs are not outputs from other neurons. This is meant to illustrate that the input layer neurons don't actually do any 'decision making' themselves, but instead have their output values set to fixed values, as a way to encode the inputs to a network. Similarly, the outputs of the neurons in the last layer of neurons, referred to as the output layer, aren't used as inputs to other neurons as they themselves are already the output of the entire network. All layers between the input and output layer are referred to as hidden layers, indicating that when interacting with a neural network, only the input and output layers are 'visible'.

2.5 Gradient Descent

Although there are other methods of learning, in this paper we shall initially focus on one of the most popular and commonly used ones in neural networks that use supervised learning: gradient descent using the backpropagation algorithm.

Previously we have mentioned that the network's parameters are adjusted while learning, based on how much the actual output of the network differs from the expected output. But how exactly should the network's parameters be adjusted?

Before making any adjustments to the network, we need to know how wrong the current output actually is, meaning we need to measure the error in the output. The individual neuron's error e_k for the k-th neuron in the output layer can herefore be defined as $e_k = y_i - d_{x,i}$. Let the output layer consist of m neurons; then, as the networks error function E, we can use the mean squared error function:

$$E = \frac{1}{2}\sum_{i=1}^{m} e_i^2 \tag{4}$$

The goal of learning is now to minimize the average output of the error function E for all inputs; the approach used in gradient descent is to reach this goal by using the error function's gradient to modify the network's parameters, effectively 'descending' to a local minimum of the error function [2].

Before going into how the gradient is defined, we shall consider how we would like to make adjustments to the network in order to reduce the error output: it is fairly intuitive that making a small modification to a bias or weight in some neuron in the network should ideally lead to a small, predictable change in the output. Here, a disadvantage of using perceptrons in a multilayer network become evident: making a small adjustment to a weight or a bias in a neuron can potentially 'flip' this neuron's output, which can lead to a knock-on effect on subsequent layers that overall has a possibly large and unpredictable effect on the ouptut. Sigmoid neurons don't have this disadvantage: it can be shown [2] that when making a small change Δw_{ij} to the j-th weight of the i-th non-input neuron and Δb_i to the bias of the i-th non-input neuron in the network, the change ΔE to the error of the total network is well approximated by:

$$\Delta E \approx \sum_{i=1}^{n} \frac{\delta E}{\delta b_i}\Delta b + \sum_{j=1}^{m_i} \frac{\delta E}{\delta w_{ij}}\Delta w_{ij} \tag{5}$$

With m_i being the amount of weights for the i-th neuron and $\delta E/\delta w_i j$ and $\delta E/\delta b$ being partial derivatives of the output regarding the weight $w_i j$ and the bias b_i, respectively. Expressed in less mathematical terms, this means that the error will only change by a small amount depending upon how 'sensitive' the network's output is to change in those parameters.

The gradient of the error function is now defined as follows:

$$\nabla E = (\frac{\delta E}{\delta b_1}, ..., \frac{\delta E}{\delta b_n}, \frac{\delta E}{\delta w_{11}}, ..., \frac{\delta E}{\delta w_{nm_n}}) \tag{6}$$

If we now define the adjustments to network's parameters as a vector of all of the individual adjustements to the network's biases and weights $\Delta p = (\Delta b_1, ..., \Delta b_n, \Delta w_{11}, ..., \Delta w_{nm_n})^T$, then we can rewrite equation 5 as:

$$\Delta E \approx \nabla E \Delta p \tag{7}$$

As we want to choose a change to the network Δp in order

doi: 10.2313/NET-2018-03-1_08

to minimize the error, we choose Δp in the following way:

$$\Delta p = -\eta \nabla E^T \tag{8}$$

We make this choice because, inserted in equation 7, this results in:

$$\Delta E \approx \nabla E \times (-\eta \nabla E^T) = -\eta ||\nabla E||^2 \tag{9}$$

As $||\nabla E||^2 > 0$, this ensures that the error function will always decrease when applying the change Δp to the network as defined above, at least within the limits of the approximation used in equation 7.

There are now three approaches for making adjustments to the network based on this method, referred to as batch learning, on-line learning and mini-batch learning.

With batch learning, we calculate the gradient ∇E for each input-output pair in the training data set and, using this, calculate the change Δp that is to be made to the parameters in the network. Then, we compute the average over all of these adjustments and finally apply these changes.

With on-line learning, we calculate the gradient ∇E and the adjustment Δp for each input-output pair individually, apply the change, and calculate the next gradient and adjustment based upon the modified network. An advantage of this method in comparison to batch learning is that it isn't necessary to store all the Δps before calculating the average; unlike with batch learning, however, it isn't possible to process the training examples in parallel, as each input-output pair of the training data is processed by a modified network based upon prior modifications.

Mini-batch learning is the compromise of the aforementioned two methods: the training data is split into equal sized 'mini batches'; for each of these mini-batches, we apply the same method used for the batch learning method. As with on-line learning, the adjustments to the network are made directly after each mini batch, with subsequent mini batches being processed by the modified network.

2.6 The Backpropagation Algorithm

So far, it has not been mentioned how the gradient ∇E required to compute the changes made to the network in the gradient descent algorithm is calculated; calculating the gradient function analytically alone for several hundred parameters, which isn't that much when compared to the dimensions found in some neural networks, isn't an option. The backpropagation algorithm presents an efficient alternative, which is why it is used for calculating the gradient required for the gradient descent algorithm.

Before continuing with the algorithm itself, however, we shall briefly introduce a new notation to make it easier to refer to individual neurons and their parameters in the network in an unambiguous way: let w_{kj}^l be the weight of the k-th neuron in the l-th layer for the output of the j-th neuron in the $(l-1)$-th layer, z_k^l the sum of the weighted inputs and the bias for the k-th neuron in the l-th layer. Analogously, b_k^l and y_k^l are defined as the bias and the output of the k-th neuron in the l-th layer.

Additionaly, we now define δ_k^l as the error function's local gradient for the k-th neuron in the l-th row:

$$\delta_k^l = \frac{\delta E}{\delta z_k^l} \tag{10}$$

In other words, a neuron's local gradient δ_k^l tells us how a change in the sum z_k^l of the neuron's inputs will affect the total error. The gradient of the error function, it can be shown, can be calculated with the help of these local gradients [2]. Specifically:

$$\frac{\delta E}{\delta w_{ik}^l} = \sigma(z_k^{l-1})\delta_j^l \tag{11}$$

and

$$\frac{\delta E}{\delta b_j^l} = \delta_j^l \tag{12}$$

The algorithm now consists of two phases, or passes: the forward pass and the backward pass. In the forward pass, we simply pass an input through the network, storing the intermediary outputs y_j^l of all neurons in the process for later use. In the second phase, we calculate the error function's local gradients δ_k^L for all neurons in the output layer, and propagate the local gradients backward throughout the network, using the local gradient $delta_k^l$ to calculate (hence the algorithm's name).

For the output layer L, the local gradients are given as:

$$\delta_j^L = \frac{\delta E}{\delta y_j^L}\sigma'(z_k^L) \tag{13}$$

Note that given our cost function $E = \frac{1}{2}\sum(y_j^L - d_{x,j})^2$, the partial derivative $\frac{\delta E}{\delta y_j^L}$ equals $(y_j^L - d_{x,i})$. Additionally, the sigma function has the practical property that $\sigma'(z_k^L) = a \times \sigma(z_k^L) \times (1 - \sigma(z_k^L)) = a \times y_k^L \times (y_k^L - 1)$, which is also one of the main reasons for it often being used as an activation function in neural networks. This allows us to rephrase equation 13 as:

$$\delta_j^L = a \times (y_j^L - d_{x,i}) \times y_j^L \times (1 - y_j^L) \tag{14}$$

As all of these components are given, we evidently have no problem calculating the local gradients for the output layer. As described above, we are now able to calculate the local gradients in the previous layer using the local gradients of the output layer. This is the formula used herefore [2]:

doi: 10.2313/NET-2018-03-1_08

$$\delta_j^l = \sum_k w_{kj}^{l+1} \delta_k^{l+1} \sigma'(z_j^l) \tag{15}$$

Once again, thanks to the aformentioned quality of the sigmoid function, this can be simplified to only contain precalculated values:

$$\delta_j^l = \sum_k w_{kj}^{l+1} \delta_k^{l+1} y_k^l \times (1 - y_k^l) \tag{16}$$

This process can now be repeated layer for layer until the local gradients have been calculated for all neurons in the network, allowing us to accurately calculate the gradient ∇E using equations 11 and 12.

3. GENETIC ALGORITHMS

Much as neural networks are based on a simplified model of how mammalian brains work, genetic algorithms are loosely based on concepts used in evolution. They are a type of optimisation algorithm, meaning that they attempt to find an optimal solution to a specific problem; this is done by modifying numerical parameters of individual solutions.

The name and the terminology used for this type of algorithm reflects it's fundamental idea's biological origin, with the main components of virtually every genetic algorithm being a way of encoding the candidate solutions as chromosomes, a fitness function for measuring the quality of such a chromosome, a selection operator and a crossover operator.

3.1 Basic Procedure

We start out with a randomized set of candidate solutions; each of these solutions is referred to as a chromosome, as has been mentioned, and the entire set is referred to as the population, with the initial population being referred to as the first generation.

In each generation, members of the population are chosen and combined in the attempt to 'breed' chromosomes with a higher fitness score; this process used to combine the chromosomes is referred to as crossover, with it's implementation varying depending on how the chromosome encodes the candidate solution and on the application of the algorithm. After a certain number of new chromosomes have been created in this way, a subset of them (usually those scoring the highest with the fitness function) replace an equal sized portion of the current population (often those scoring low with the fitness function).

This process is then repeated generation for generation, with the population staying a constant size, until either a certain number of generations have passed, or a chromosome scores higher than a predefined value for the fitness function; the algorithm then returns the fittest chromosome as the optimal solution [5].

3.2 Chromosome Encoding

A central question when applying a genetic algorithm to a problem is how to encode the candidate solution as a chromosome, as this has a direct impact on the crossover operation. Principally, a chromosome hasn't got any practical restrictions on how it can be encoded: one approach, for example, is to encode the candidate solution as an array of strings, while another might be to store it as a series of integer values or bits. As with all aspects of genetic algorithms, the exact implementation depends on the problem and the space of candidate solutions.

1	0	1	0	1	1	0	0	1	0

Figure 2: An example chromosome using binary encoding

A simple example would be encoding candidate solutions for the well known knapsack problem: one is presented with a selection of $n \in \mathbb{N}$ items, each with a weight w_i and a value v_i, and at one's disposal is a bag that can handle a certain maximum weight w_max. The problem now is to find a combination of items that maximize the value that one can carry in the bag.

In this case, a candidate solution could be encoded as a vector of bits c, with a 1 at the i-th position meaning that the i-th item should be included in the bag, and a 0 meaning it shouldn't. A chromosome encoding the candidate solution of including the 1st, 3rd, 5th, 6th and 9th item out of 10 items can be seen in figure 2.

An appropriate fitness function f could therefore be defined as:

$$f(c) = \begin{cases} v^T c & \text{if } w^T c \leq w_{max} \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

Note that we could remove the condition in the fitness function if we didn't allow chromosomes representing items with a total weight larger than w_{max} to exist in the first place; as this is just meant to be an example, however, this way of defining the fitness function is perfectly sufficient.

3.3 Crossover and Mutation

Crossover operations can be as varied as the different possibilities for encoding the chromosomes, and their exact workings will be closely linked to the chromosomes' form. To gain some insight into how such operators may work, we shall consider two reasonably simple methods: single-point and multi-point crossover.

For the single-point crossover operation, we select a single point in the chromosome's encoding, and generate the child by using the parameters of the first parent chromosome until that point, and using the parameters of the second parent from that point onward. To return to our previously used example of the binary encoding of the knapsack problem, this can be seen in figure 3.

The mullti-point crossover operation is defined similarly, but instead of using one point, we use several instead; once again

doi: 10.2313/NET-2018-03-1_08

Parent 1:	1	0	1	0	1	1	0	0	1	0
Parent 2:	1	1	0	0	1	0	0	1	1	1
⇓ Crossover ⇓										
Child 1::	1	0	1	0	1	0	0	1	1	1
Child 2:	1	1	0	0	1	1	0	0	1	0

Figure 3: An example of single-point crossover

Parent 1:	1	0	1	0	1	1	0	0	1	0
Parent 2:	1	1	0	0	1	0	0	1	1	1
⇓ Crossover ⇓										
Child 1:	1	0	0	0	1	1	0	1	1	0
Child 2:	1	1	1	0	1	0	0	0	1	1

Figure 4: An example of multi-point crossover

using the knapsack example, a multi-point crossover with 4 points is visualized in figure 4.

Usually, after the child chromosome has been generated, there is a chance of of the new chromosome 'mutating' new properties; this involves randomly changing one or more parameters of the chromosome, within the limits of the solution space. For a bit-sequence chromosome like the one in our example, this could mean randomly selecting one or more bits and flipping them. This element of randomness is meant to add 'genetic diversity' to the population so that the algorithm doesn't get stuck in non-optimal local maxima of the fitness function as easily [5].

3.4 Selection

In order to perform the crossover operation, we have to first select two parents; for this, we have to define a selection operator that selects two parents from our population. Ideally, we would like to 'breed' favourable qualities for our next generation of chromosomes, which are usually contained in the chromosomes with the best fitness scores. One way of achieving this is to define a selection operator that selects population members randomly, with the probability $P(c_i)$ of each chromosome c_i of a population of size n being picked being defined as:

$$P(c_i) = \frac{f(c_i)}{\sum_{j=1}^{n} f(c_j)} \tag{18}$$

Effectively, this means that chromosomes with higher fitness score will be preferred during selection, ensuring that favourable qualities are more likely to be contained in children created by the crossover operation.

Towards the end of the algorithm's run, however, the fitness values will only vary slightly, meaning that all members of a population have roughly equal chances of being selected; also, it has the disadvantage is that it only works for fitness functions for which higher outputs are defined as being better.

An alternative approach that avoids these restrictions may instead work by selecting chromosomes based upon the rank of their fitness score among the population, instead of based on their relative fitness score [6]. For a population of size n, the chromosome c_i with rank $i \in [1; n]$ regarding it's fitness score, the probability $P(c_i)$ could then be defined in the following way:

$$P(c_i) = \frac{2(n - i + 1)}{n(n + 1)} \tag{19}$$

4. EVOLVING NEURAL NETWORKS

Normally when designing a neural network, the implementer has to make a series of design choices: these could include, for example, the network's topology, the backpropagation algorithm's learning rate and the activation function used by the neurons in the network; these properties are sometimes referred to as a network's hyperparameters[2].

Although there are empirical results and heuristics that give some general rules of thumb that aid in those decisions, their exact effect on performance is generally not so well understood; this means that finding hyperparameters that work well usually involves some amount of trial and error. An alternative approach to manual selection is to use some form of optimisation algorithm to determine some of these parameters; thus, roughly since the 1980s, a good deal of effort has gone into combining genetic algorithms with neural networks for this purpose [6].

There are primarily three ways in which genetic algorithms have successfully been combined with neural networks: evolving the weights of a network with fixed hyperparameters, evolving a networks hyperparameters, and evolving a network's topology alongside it's weights [10]. In this paper, we shall consider the latter two.

4.1 Encoding Networks

Before getting to the actual ways in which we can use genetic algorithms to evolve neural networks, it is relevant to bring to the reader's attention some problems that arise when attempting to actually implement this.

As has been mentioned already, for genetic algorithms, a candidate solution's parameters have to be encoded in a chromsome. One problem one encounters when encoding a neural network is recurred to as the competing conventions problem: in essence, several networks that are functionally identical can be structurally different [7]. As a simple example, assume we naively encode the parameters $params_i$ of n of n individual neurons in a network sequentially:

$params_1$	$params_2$	...	$params_n$

We shall assume for this example that the parameters include the weights, the biases and the layer of each neuron, and that the neurons are sorted incrementally by their layer, meaning that the input layer neurons are encoded at the beginning and the output layer neurons are encoded at the end. If we consider a simple network with only one hidden layer with m neurons using this form of encoding, then there

doi: 10.2313/NET-2018-03-1_08

are $m!$ possible permutations for the hidden layer and, consequently, $m!$ different representations for functionally the same network.

This may not be seem to be an immediate problem, and there is in fact research that suggests that it may not have a significant impact on the genetic algorithm [6]. On the other hand, crossovers of functionally similar, but structurally different parent networks tend to result in impaired children, which needlessly cost the algorithm additional computation time [7].

Another complication occurs with larger networks: the solution space that the genetic algorithms have to search tend to simply have too many dimensions in which optimisation is possible [6]. When focusing merely on the hyperparameters of the network, however, it has been shown that even for more complex so-called deep neural networks, which use many hidden layers in their topology, it is possible to autonomously evolve state of the art hyperparameters using genetic algorithms[11].

4.2 Hyperparameter Optimisation

One way of using a genetic algorithm to select hyperparameters for a neural network lies in encoding merely a network's hyperparameters as a chromosome, with the networks' weights and biases not being encoded in the chromosomes; this is referred to as Baldwin learning [6]. The exact hyperparameters that are included may vary strongly depending on the implementation: using feedforward neural networks as described previously in this paper, this could only involve encoding the number of hidden layers used and the amount of neurons in each of these layers, with some fixed maximum amount of hidden layers [10], or additionally include the activation function, the learning rate η of the backpropagation algorithm and even the learning algorithm used [9].

As part of the evaluation by the fitness function, we first initialize a network with the encoded hyperparameters, with randomized starting weights and biases, and then train it. After finishing it's training, we test it's performance on a set of test data that wasn't used during training, using, for example, the mean square error of all outputs of the test data as the output of the fitness function.

Unfortunately, the process of training just a single generation, which can have tens or even hundreds of chromosomes, is very computationally expensive, as even training a single network with backpropagation can potentially take days. As a sufficiently well performing set of hyperparameters can theoretically take many generations to evolve; this means that even with a high degree of parallelization, e.g. training all chromosomes of a generation simultaneously, this process can take a large amount of time.

Interestingly, the choice of the amount of training that a network undergoes before evaluation has been shown to have a direct impact on the genetic algorithm's evolution rate [8], as well as how fast networks that are produced in this manner tend to learn [11].

Overall it can be summarized that through applying appropriate limitations to the search space of the genetic algorithms, such methods should be a very effective alternative to manually selecting the hyperparameters of a network, even if the computational cost may still often be significant.

4.3 Neuroevolution

Neuroevolution, as the name implies, is a description for methods in which evolutionary algorithms, of which genetic algorithms are a subtype, are used to train neural networks. When considering such methods, we differentiate between fixed topology systems, and so-called Topology and Weight Evolving Artificial Neural Networks, or TWEANNS [13]. Evolving the network's weights has the advantage that it isn't necessary to calculate the error function gradient to make adjustments (as opposed to gradient descent); this makes neuroevolution a good alternative to using gradient descent when the gradient is either very expensive, or not possible to calculate for a given problem [12]. On the other hand, as was already mentioned, the fact that all of the network's parameters are encoded in the genetic algorithm's chromosome means that training networks this way isn't really an option for larger networks [6].

When evolving the topology alongside the weights, an additional problem arises: when a chromosome evolves with an adjusted structure, this adjustment will often initially have a negative impact. As an example: by adding a neuron with randomly initiated weights at a critical point in the network, we could completely change the way the network functions, potentially 'ruining' a network that performed comparatively well beforehand. This will lead to a lower fitness score, which may well in turn lead to the new network structure disappearing from the population's 'gene pool', even if the new structure might perform far better than the old structure after correctly evolving the appropriate weights. This problem can usually be addressed by having networks only compete with similarly-structured networks, referred to as speciation [13]. This, on the other hand, means that a computationally affordable way of categorizing two networks as having a 'similar structure' in this context needs to be provided [7].

As an example of a TWEANN, we can consider the NeuroEvolution of Augmenting Topologies algorithm, referred to as NEAT [13]. The approach used attempts to keep the dimensionality of the algorithm's search space minimal by starting with networks of minimal structural complexity (i.e. only an input and output layer), and gradually incrementing the structure; mutations, it is worth noting, are only capable of adding structure, not removing it. Specifically, the first generation of chromosomes only consists of networks with an input and an output layer, and additional structure is evolved by adding neurons and connections between neurons through mutation; this technique is referred to as complexification [13]. When compared to methods that involve initial populations of random structure, complexification tends to result in neural networks with a comparatively minimal structure. This is favourable as a smaller structure means less computational effort to calculate the output for a given input, and, depending on the network encoding scheme, less storage space required for the network. Additionally, due to the structure being kept minimal, the dimensionality of the search space of the genetic algorithm is also relatively low.

doi: 10.2313/NET-2018-03-1_08

This benefits the performance of the algorithm [7]. Note that due to the way that structure is added, depending on the actual implementation, feedforward networks aren't the only type of network that can be evolved in this way as the network can also include loops.

In addition to complexification, NEAT features an encoding scheme that allows an efficient structural comparison between networks, enabling the protection of innovative structure through the aforementioned method of speciation. Chromosomes consist of node genes and connection genes, each of which is assigned a unique historical marker when evolved through mutation, with genes passed on to offspring through crossover using the same marker. Concerning the protection of topological innovations through speciation, NEAT uses a method referred to as explicit fitness sharing. By checking which genes overlap in two different chromosomes i and j, we can assign a distance $\delta(i,j)$ to these two networks, with a higher distance meaning a greater structural difference. After evaluating the fitness f_i of a chromosome i in a population of n chromosomes, it is assigned it's modified fitness f_i' in the following way:

$$f_i' = \frac{f_i}{\sum_{j=1}^{n} sh(\delta(i,j))} \tag{20}$$

With the share function sh being defined as a step function that outputs 0 if the structural distance between i and j is above some specified limit, and 1 otherwise. An output of 1 is thus to be interpreted as chromosome i and j belonging to the same species; if there are many chromosomes in a species, then they're modified fitness f' will be comparatively lower, while the reverse is true for species with few chromosomes. This means that new species that emerge, whether through mutation or crossover, will be assigned a higher fitness score if they structurally differ from other species sufficiently. Using this modified fitness function, each species is implicitly allocated a slot, or evolutionary niche, in the population based upon their fitness, with the weakest members of each species having high chances of being eliminated in every generation. [13].

Neuroevolutionary methods, as a whole, can be seen as a good alternative to gradient descent for many problems for which calculating the error function gradient isn't an option; also, regardless of whether the topology is fixed or not, due to the mutative qualities of genetic algorithms, it is less likely that the optimisation process will stall in a suboptimal local performance maximum [13]. Also, due to the nature of genetic algorithms, much of the computationally most expensive phase, namely that of the fitness evaluation, can be performed in parallel, enabling a great increase on performance [7]. TWEANNS carry with them the additional advantage that they relieve the programmer implementing the algorithm of having to make exact choices concerning the network's topology. The main disadvantage of using neuroevolution is the limit in the size of the networks: if the networks become too complex, then evolutionary algorithms will struggle to find fitting parameters due to the high dimensionality of the search space [6].

5. CONCLUSION

We have seen that neural networks and genetic algorithms are two concepts with biological backgrounds that synergize well for many problems: as long as the dimensionality of the search space isn't too big, genetic algorithms are capable of finding good parameters for many applications of neural networks, be it the hyperparameters of the network, the parameters of the individual neurons, or both. Due to the amount of attention that neural networks are receiving of late, and with the diversity of methods in neuroevolution and hyperparameter optimisation using genetic algorithms, future advances in this field don't seem unlikely, with new methods possibly surpassing some of the limitations that are present today.

6. REFERENCES

[1] F.A. Azevedo, L.R. Carvalho, L.T. Grinberg, et al.: *"Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain"*, The Journal of Comparative Neurology, 513 (5), pages 532-41, 2009

[2] Michael A. Nielsen: *Neural Networks and Deep Learning*, Determination Press, 2015

[3] Alan F. Murray: *Applications of Neural Networks*, pages 1-4, Springer, Boston, MA, USA, 1995

[4] Simon Haykins: *Neural Networks - A Comprehensive Foundation*, Pearson Education, Inc., 3rd edition, pages 65-69 and 157-172, 2009

[5] Jenna Carr: *An Introduction to Genetic Algorithms*, 2014

[6] Phillip Koehn: *"Combining Genetic Algorithms and Neural Networks: The Encoding Problem*, University of Tennesy, Knocksville, 1994

[7] William T. Kearney: *"Using Genetic Algorithms to Evolve Artificial Neural Networks"*, Colby College, 2016

[8] R. Keesing and D.G. Stork: *"Evolution and Learning in Neural Networks: the Number and Distribution of Learning Trials Affect the Rate of Evolution"*, Proceedings of the 6. Neural Information Processing Systems Conference, pages 804-810, 1993

[9] O.A. Abdalla, A.O. Elfaki, Y.M. AlMurtadha: *"Optimizing the Multilayer Feed-Forward Artificial Neural Networks Architecture and Training Parameters using Genetic Algorithm"*, International Journal of Computer Applications (0975-8887)2014

[10] Christopher M. Taylor: *"Selecting Neural Network Topologies: A Hybrid Approach Combining Genetic Algorithms and Neural Networks"*, Southwest Missouri State University, 1997

[11] Risto Miikkulainen et. al: *"Evolving Deep Neural Networks"*, Sentient Technologies, Inc. / The University of Texas at Austin, 2017

[12] A.J. Turner, J.F. Miller: *"The Importance of Topology Evolution in NeuroEvolution: A Case Study Using Cartesian Genetic Programming of Artificial Neural Networks"*, Research and Development in Intelligent Systems XXX, pages 213-226, Springer International Publishing Switzerland, 2013

[13] K.O. Stanley: *"Efficient Evolution of Neural Networks through Complexification"*, pages 7-41, The University of Texas at Austin, 2004

ISBN 978-3-937201-61-0
DOI 10.2313/NET-2018-03-1
ISSN 1868-2634 (print)
ISSN 1868-2642 (electronic)