Network Architectures
and Services
NET 2014-08-1

FI & IITM
SS 2014

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

Summer Semester 2014

Munich, Germany, 30.04.2014-03.08.2014

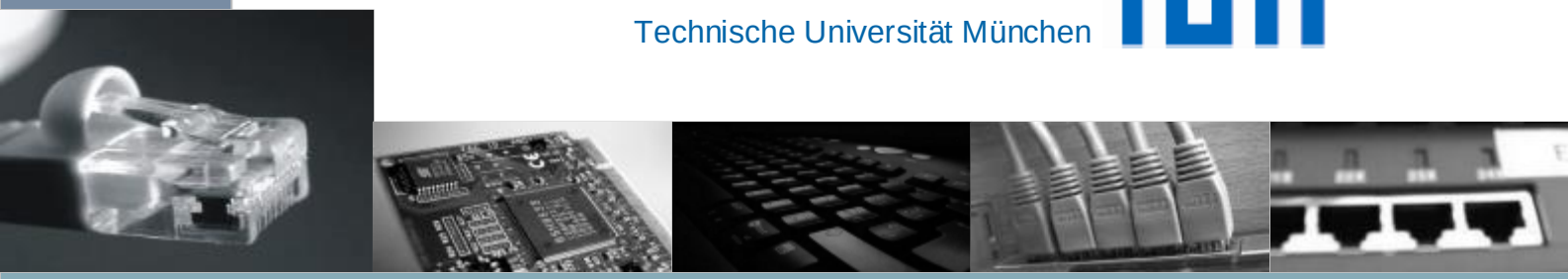
Editors

Georg Carle, Daniel Raumer, Lukas Schwaighofer

Organisation

Chair for Network Architectures and Services
Department of Computer Science, Technische Universität München

Technische Universität München



Network Architectures
and Services
NET 2014-08-1

FI & IITM
SS 2014

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

Sommersemester 2014

München, 30. 04. 2014 - 03. 08. 2014

Editoren: Georg Carle, Daniel Raumer, Lukas Schwaighofer

Organisiert durch den Lehrstuhl Netzarchitekturen und Netzdienste (I8),
Fakultät für Informatik, Technische Universität München

Technische Universität München

Proceedings of the Seminars
Future Internet (FI), and Innovative Internet Technologies and Mobile Communication Networks (IITM)
Summer Semester 2014

Editors:

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

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

Lukas Schwaighofer
Lehrstuhl Netzarchitekturen und Netzdienste (I8)
E-mail: schwaighofer@net.in.tum.de
Internet: `http://www.net.in.tum.de/~schwaighofer/`

Cataloging-in-Publication Data

Seminars FI & IITM SS 2014
Proceedings zu den Seminaren „Future Internet“ (FI) und „Innovative Internettechnologien und Mobilkkommunikation“ (IITM)
München, Germany, 30. 04. 2014 - 03. 08. 2014
ISBN: 3-937201-43-2

ISSN: 1868-2634 (print)
ISSN: 1868-2642 (electronic)
DOI: 10.2313/NET-2014-08-1
Lerhstuhl Netzarchitekturen und Netzdienste (I8) NET 2014-08-1
Series Editor: Georg Carle, Technische Universität München, Germany
© 2014, Technische Universität München, Germany

Vorwort

Wir präsentieren Ihnen hiermit die Proceedings zu den Seminaren „Future Internet“ (FI) und „Innovative Internettechnologien und Mobilkommunikation“ (IITM), die im Sommersemester 2014 an der Fakultät für Informatik der Technischen Universität München stattfanden. Den Teilnehmerinnen und Teilnehmern stand es für das Paper und für den Vortrag jeweils frei, die englische oder die deutsche Sprache zu benutzen. Dementsprechend finden sich sowohl englische als auch deutsche Paper in diesen Proceedings.

Erstmals verliehen wir beginnend mit dem IITM Seminar den Best Paper Award. Er ging an Herrn Tobias Klenze, der mit seiner Seminararbeit zum Thema „Designing privacy preserving systems“ auf gute und strukturierte Weise einen Einblick in die Herausforderungen und den Aufbau privacy erhaltender Softwaresysteme bietet. In seiner Arbeit gibt er auch eigene Vorschläge zur Verbesserung vorhandener Ansätze.

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

Im Seminar FI wurden Beiträge zu aktuellen Themen der Netzwerkforschung vorgestellt. Die folgenden Themen wurden abgedeckt:

- Bootstrapping Verfahren für Peer-to-Peer VPN
- Management von verteilten Netzwerkexperimenten mit NEPI
- Methoden der Botnetzerkennung
- Botnetzerkennung mittels DNS
- Einsatz der GPU zur Netzwerkpaketverarbeitung
- Techniken zur schnellen Paketverarbeitung in PC-Systemen
- Hohe Zuverlässigkeit von IT-Systemen und deren Fehler
- Anonymität mit dem Mobiltelefon in der Menge
- Sicherheit im Umfeld intelligenter Gebäude

Auf `http://media.net.in.tum.de/#%23Future%20Internet%23SS14` können die aufgezeichneten Vorträge zu diesem Seminar abgerufen werden.

Im Seminar IITM wurden Vorträge aus dem Themenbereich der Netzwerktechnologien inklusive Mobilkommunikationsnetze vorgestellt. Die folgenden Themen wurden abgedeckt:

- Deterministische Ethernet-Flows für Industrienetzwerke
- Performanzanalyse von TCP Flows
- Formale Header Space Analyse
- SDN Northbound API: Anforderungen und Trends
- Namecoin
- Intels Dataplane Development Kit (DPDK)
- Smart Metering und Privatsphäre
- Privatsphäre bei Websuchen
- Privatsphäre von DNS

- Evil-Twin-Accesspoint Angriff
- Sicherheit von Open-Source Software
- Missbrauchserkennung in Sozialen Netzwerken
- Selbstheilung von selbstorganisierenden Netzwerken

Auf `http://media.net.in.tum.de/#%23IITM%23SS14` 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 habe, so finden Sie weitere Informationen auf unserer Homepage `http://www.net.in.tum.de`.

München, August 2014

Georg Carle

Daniel Raumer

Lukas Schwaighofer

Preface

We are pleased to present you the interesting program of our main seminars on "Future Internet" (FI) and "Innovative Internet Technologies and Mobil Communication" (IITM) which took place in the summer semester 2014. In both seminar courses the authors were free to write their paper and give their talk in English or German.

Starting with the IITM seminar we have decided to honor the best paper with an award. The first Best Paper Award was given to Mr Tobias Klenze for his paper on "Designing privacy preserving systems". He gave an insight into the challenges and design of privacy preserving systems and expanded on existing approaches with his own ideas.

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

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

- Bootstrapping Peer-to-Peer VPN
- Network experiments with NEPI
- Survey of botnet detection methods
- Botnet detection with DNS
- GPU based network packet processing
- Survey of high speed packet processing techniques in PC-systems
- High reliability and when things go wrong
- Privacy of mobile phones in the crowd
- Security in smart buildings

Recordings can be accessed on `http://media.net.in.tum.de/#%23Future%20Internet%23SS14`.

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

- Determinism of Ethernet flows in industrial networks
- Performance evaluation of TCP flows
- Formal header space analysis
- Northbound API Requirements for SDN
- Namecoin
- Intel's Dataplane Development Kit (DPDK)
- Smart Metering and Privacy
- Privacy and web search
- DNS Privacy
- Evil-Twin-Accesspoint attacks
- Security analysis of Open-Source software

- Abuse detection
- Self-Healing in Self-Organizing Networks

Recordings can be accessed on `http://media.net.in.tum.de/#%23IITM%23SS14`.

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

Munich, August 2014

Seminarveranstalter

Lehrstuhlinhaber

Georg Carle, Technische Universität München, Germany (I8)

Seminarleitung

Daniel Raumer, Technische Universität München, Germany
Lukas Schwaighofer, Technische Universität München, Germany

Betreuer

Cornelius Diekmann (diekmann@net.in.tum.de)
Technische Universität München, Mitarbeiter I8

Oliver Gasser (gasser@net.in.tum.de)
Technische Universität München, Mitarbeiter I8

Fabien Geyer (fgeyer@net.in.tum.de)
Technische Universität München,
Externer Doktorand

Benjamin Hof (hof@net.in.tum.de)
Technische Universität München, Mitarbeiter I8

Holger Kinkelin (kinkelin@net.in.tum.de)
Technische Universität München, Mitarbeiter I8

Marcel von Maltitz (maltitz@net.in.tum.de)
Technische Universität München, Mitarbeiter I8

Johannes Naab (naab@net.in.tum.de)
Technische Universität München, Mitarbeiter I8

Heiko Niedermayer (niedermayer@net.in.tum.de)
Technische Universität München, Mitarbeiter I8

Daniel Raumer (raumer@net.in.tum.de)
Technische Universität München, Mitarbeiter I8

Lukas Schwaighofer (schwaigh@in.tum.de)
Technische Universität München, Mitarbeiter I8

Tsvetko Tsvetkov (tsvetkov@net.in.tum.de)
Technische Universität München, Mitarbeiter I8

Matthias Wachs (wachs@net.in.tum.de)
Technische Universität München, Mitarbeiter I8

Florian Wohlfart (wohlfart@net.in.tum.de)
Technische Universität München, Mitarbeiter I8

Seminarhompage

`http://www.net.in.tum.de/de/lehre/ss14/seminare/`

Inhaltsverzeichnis

Seminar Future Internet

Session 1: Innovative Peer-to-Peer

Bootstrapping P2P VPN 1
Felix Weißl (Betreuer: Benjamin Hof, Lukas Schwaighofer)
Netzwerkexperimente mit NEPI 9
Tobias Betz (Betreuer: Matthias Wachs)
Überblick über Botnetz-Erkennungsmethoden 17
Florian Zipperle (Betreuer: Oliver Gasser)
Botnet Detection with DNS Monitoring 25
Christopher Will (Betreuer: Oliver Gasser)

Session 2: Deadling with High Speed Networks

Using GPUs for Packet-processing 33
Thomas Schultz (Betreuer: Daniel Raumer)
A Survey of Trends in Fast Packet Processing 41
Konstantina Tsiamoura (Betreuer: Florian Wohlfart, Daniel Raumer)

Session 3: Secure & Private in the Internet of Things

Internet Science – High Reliability or When Things go wrong and as things change 49
Tilman Eberspächer (Betreuer: Heiko Niedermayer)
What does your mobile phone leak – Unique in the crowd and predictable 57
Matti Lorenzen (Betreuer: Heiko Niedermayer)
Survey: Security in Smart Building Networks 65
Michaela Mohl (Betreuer: Holger Kinkelin)

Seminar Innovative Internet Technologien und Mobilkommunikation

Session 1: Formal Methods for Network Analysis

Determinism for Ethernet flows in industrial networks . . . 73
Lukas Grillmayer (Betreuer: Fabien Geyer)
Performance evaluation of TCP flows . . . 79
Florian Hisch (Betreuer: Fabien Geyer)
Formal Methods: Header Space Analysis . . . 89
Benedikt Jaeger (Betreuer: Cornelius Diekmann)

Session 2: Innovative Trends and Technologies

Northbound API Requirements for SDN . . . 97
Christian Fuchs (Betreuer: Daniel Raumer)
Namecoin . . . 107
Andreas Loibl (Betreuer: Johannes Naab)
A Look at Intel's Dataplane Development Kit . . . 115
Dominik Scholz (Betreuer: Daniel Raumer, Florian Wohlfart)

Session 3: Privacy

Privacy Strategies in Smart Metering . . . 123
Tobias Klenze (Betreuer: Benjamin Hof)
Der Wert von Privatsphäre bei Websuchen und Personalisierung von Websuchen . . . 133
Sebastian Vogl (Betreuer: Heiko Niedermayer)
DNS Privacy . . . 141
Patrick Werneck (Betreuer: Johannes Naab)

Session 4: Security Vulnerabilities and Countermeasures

Evil-Twin-Accesspoint . . . 149
Andreas Huber (Betreuer: Marcel von Maltitz)
Analyse zur Sicherheit von Open-Source Software . . . 159
Marcus Rogowsky (Betreuer: Heiko Niedermayer)
Abuse Detection . . . 167
Peter Gawronski (Betreuer: Marcel von Maltitz)
Self-Healing in Self-Organizing Networks . . . 175
Oliver Scheit (Betreuer: Tsvetko Tsvetkov)

Bootstrapping P2P VPN

Felix Weißl
Betreuer: Benjamin Hof, Lukas Schwaighofer
Seminar Future Internet SS2014
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: felix.weissl@tum.de

ABSTRACT
Decentralized or P2P (peer-to-peer) VPNs (virtual private networks) are popular due to their advantage over centralized VPNs or the classic client-server model. This paper surveys the main difficulties of P2P VPNs in general, but most importantly of bootstrapping a new peer in such a network which is the mechanism of a new peer finding and joining it. Different solutions are indicated on existing implementations and evaluated separately. Especially the two contrasting models of supernodes versus the joining over an existing public overlay are examined for they are most commonly deployed in existing P2P VPNs. It becomes apparent that the presented solutions only partly cover the needed functionality, especially in terms of security and the traversal of NAT devices.

Keywords
peer-to-peer, virtual private network, boostrapping, decentralized overlay, NAT traversal

1. INTRODUCTION
Peer-to-peer networking has gained a great interest in the last years due to many popular applications in file sharing (BitTorrent, Gnutella, KaZaA), but also media streaming or voice-over-IP services such as Skype. Its main advantage over the client-server model is that no central authority such as a server is needed for sharing resources because every node (peer) is taking part in the network actively, by meaning of relaying data for other nodes.

A virtual private network (VPN) provides an illusion of a local area network by creating secure and authenticated communication links amongst its participating nodes over the global Internet [23]. Today VPNs are frequently used, e.g. in companies so that employees can enter the intranet securely without beeing physically linked to it. But they are also attractive to home users who can establish a secure end-to-end connection with each other for sharing data or other local resources.

A P2P VPN - as the name suggests - combines these concepts and therefore offers a completely decentralized overlay network. Operations that work for client-server model have to be rethought and adapted to a distributed overlay model.

The rest of this paper is organized as follows. Section 2 gives short background information and definitions about the main terms used later in this work. Section 3 explains the key requirements of a P2P VPN and two main problems of the bootstrapping mechanism: peer discovery and NAT traversal. Existing implementations and how they solve these issues are discussed in Section 4. Other interesting or notable approaches are shown in Section 5 while Section 6 presents concluding remarks.

2. BACKGROUND
A P2P system relies on a network topology where the nodes are connected to each other and have the purpose of sharing resources such as storage, bandwith or CPU cycles. They form a self-organizing overlay without the need of a centralized server or authority [18, p. 6].

An *overlay* network is defined as an application layer virtual or logical network in which end points or nodes are addressable and that provides connectivity, routing and messaging between these nodes [18, p. 7]. An overlay is layered on top of an existing layer, called underlay, which in our context is the global Internet [3].

In *centralized* VPNs all the communication flows through one server which also provides bootstrapping and authentication. Two peers never directly send packets to each other because there is always a central authority such as a dedicated server relaying the traffic between them. Of course, this server can become a single point of failure which is a problem for high availability networks. A popular implementation of this technique is OpenVPN [15].

In *decentralized* networks however, there is no distinction between clients and servers since every peer in the network is equal. Routing, bootstrapping and authentication is done directly with each other and as a result must be explicitly defined [23]. By decentralizing the network, bandwidth and computation can be distributed between the peers.

In *unstructured* P2P systems a node relies only on its adjacent nodes by the terms of routing and forwarding. The graph is randomly created depending on which nodes are joining or leaving and their interactions. That makes joining and leaving rather easy, but has a problem in searching, e.g. a file, because the request has to go through the entire network in the worst case. Two popular implementations are n2n [13] and P2PVPN [6] which are discussed in Section 4.1.

Structured P2P systems provide distributed lookup services by forming a self-organizing topology such as a ring [23],

doi: 10.2313/NET-2014-08-1_01

a tree or a mesh structure where every node has a link to all other nodes in the network. These lookup services are provided by so called *distributed hash tables (DHT)*. They are decentralized data structures, in which keys with associated data are mapped to specific node IDs in the overlay [21]. DHTs are typical for structured P2P systems and commonly used in P2P architectures.

3. REQUIREMENTS AND DIFFICULTIES

Before introducing existing P2P VPN implementations this Section first mentions certain requirements that have to be fulfilled and difficulties that have to be dealt with.

3.1 Requirements

The main design goals mentioned in [8, 9, 2] for a purely decentralized bootstrapping mechanism to fullfil are:

1. Availability: Every peer must be able to join or leave the network when it desires to. That means the bootstrapping mechanism must be guaranteed at any time which makes an approach with an unstable success rate insufficient [8]. A single point of failure should always be avoided as far as it is possible for decentralized networks. That is also ensured by equally distributing workload between the peers so that an attacker cannot compromise the availybility by initiating DDoS attacks on them.

2. Automation and Self-organization: For a desired real world applicability a peer has to be able to join the network with as little as possible manual interaction. Furthermore the bootstrapping mechanism should not have to rely on prior knowledge, e.g. a list of bootstrap servers that a joining node needs to have stored [2].

3. Efficiency: Although our main focus in this paper is the connection establishment, it is important to keep the time period for that to a minimum.

4. Scalability and Robustness: The bootstrapping mechanism has to work in small private as well as in large communication networks involving millions of peers. If certain peers malfunction or leave the network rapidly, there have to be procedures changing the network back to its full functionality. In this context "churn" is the rate of joining and leaving peers of a network [1].

5. Security: Trusting each other in P2P systems is harder than in centralized systems where authorities can interact as a trusting party such as a PKI. Therefore is important to prevent bad peers from joining the network.

3.2 Finding a peer

The initial problem for a peer before connecting to the P2P VPN is to discover the network and an active participant as an entry point. We assume that a previous connection has never been established so it is not possible to own some kind of peer cache.

As a first approach one could integrate a list of known peers into the P2P software itself. Gnutella for instance, a very large decentralized peer-to-peer network, uses this approach [5]. Of course these peers must have a high availability and become target for attacks. Internet providers could use this information to prevent their clients from joining such a network. The peer list, in this context often called hostlist or webcache (Gnutella) [7], could also be shared and requested on a public HTTP server. But this would end up in a partially centralized architecture, so this is not an optimal solution. The second approach demonstrated in [5] uses brute-force scanning over a list of promising IP addresses. The list is a result of a special heuristic using statistical profiles using the IP ranges of start-of-authorities in the domain name system [5]. This method requires that developers can get a list of IP addresses for the network and that the peers use the default port for the protocol. A similar approach is presented in Section 5.

3.3 Middlebox Traversal

Another technical issue with P2P VPNs are middleboxes such as *Network Address Translators* (NATs) and *firewalls*. A firewall is used to filter incoming and outgoing traffic and is set up on the edge of the global Internet to a local network for protection. The firewall's filter rules are usually designed to block incoming connections that are not a response to a previous outgoing request. That is a problem for P2P networks where a new peer wants to communicate with another peer behind such a device for bootstrapping. NAT devices on the other hand help to overcome the shortage of global IPv4 addresses by hiding a local network behind just one globally routable IP address. All clients in this local network are given private IP addresses (e.g. in 192.168.0.0/16) and every packet's local IP and port are translated to a global IP and port. That means that local clients cannot be addressed directly with a unique IP before the peer behind the NAT initiates the connection and the NAT can route the packet properly. The ongoing migration from IPv4 to IPv6 does not necessarily resolve this issue. Although NAT is not needed anymore and peers could directly communicate, firewalls would still be essential for blocking unwanted, incoming connections. Also since IPv4 is still deployed, NAT will be as well.

It can be assumed that the majority of peers in P2P networks are home users and behind a NAT device or a firewall as well. Thus *NAT traversal* mechanisms are inevitable for operating a P2P VPN.
A simple mechanism for a host behind a NAT ("NATed" host) would be to configure the NAT device yourself and enable port forwarding, so that all connections can traverse on a negotiated port. Because the majority of users would not have the rights or knowledge to do this, this is not really an option. The common NAT traversal mechanisms used in current implementations are elaborated on the subsequent Sections.

3.3.1 TURN: Traversal using Relay around NAT

TURN [10] is very reliable and deployed in centralized VPNs. If two NATed hosts want to communicate, each of them has to initiate a connection with a well-known globally routable server. This TURN server relays every message between both hosts and therefore also passing both NAT devices. Of course this is a very inefficient and resource-intensive method and only scalable for small networks. But since this

doi: 10.2313/NET-2014-08-1_01

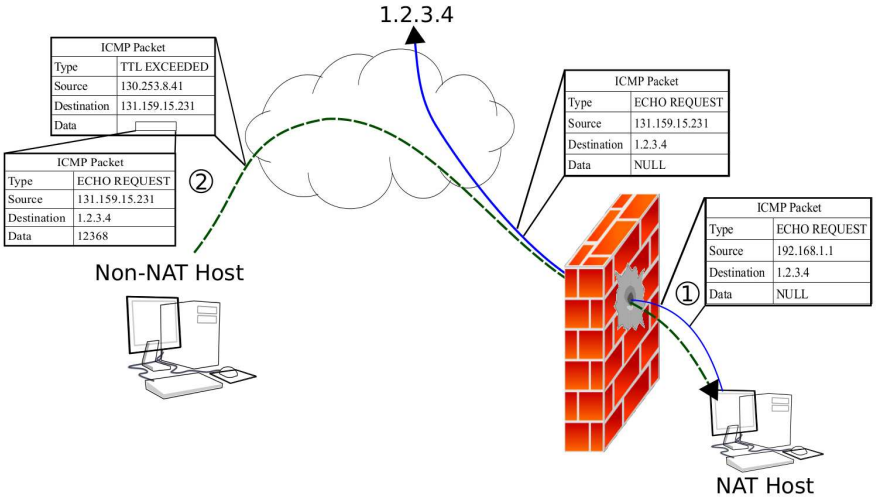


Figure 1: Here *autonomous traversal* of a NAT device (represented as wall) is shown. The solid line describes outgoing and the dashed line incoming ICMP packets. Figure source: [11]

is the only method working on all existing NATs relaying is a useful alternative if other traversal methods fail [4].

3.3.2 UDP Hole Punching

"Hole punching" is one of the simplest and most practical NAT traversal techniques [4] for the scenario that two hosts are behind a NAT device. Both NATed hosts have to have an active UDP session with a so-called rendezvous server. First both hosts exchange their private and public endpoint information (local and global IP:Port) via the rendezvous server as an intermediary using the Session Traversal Utilities for NAT protocol (STUN) [16]. By this point the server is not needed anymore. Both hosts try to establish a UDP connection between each other on both endpoints. Considering the simple scenario of both hosts residing in the same network, the connection can be established. In the scenario with two different NATs, the first outgoing message of both hosts "punches a hole" through their own NATs. That means if the NAT is well-behaved, it will preserve the hosts private endpoint and allow incoming traffic and address translation [4]. Well-behaved NATs in this context are *cone* NATs, which support consistent endpoint translation, and not *symmetric* NATs, where "hole punching" is not successful [4]. Tests on a wide variety of deployed NATs have shown that about 82% support hole punching using UDP [4].

3.3.3 TCP Hole Punching

Since there are VPNs relying on TCP such as P2PVPN (Section 4.2.1) hole punching is needed for TCP as well. Compared to UDP the idea is very similar, however since TCP is a stateful protocol, there are some issues. For a successful hole punching both NATs need to be able to establish a connection with the rendezvous server and the other hosts NAT device over the same port. Both connections need a TCP socket which binds to the specific port and this fails if the port is already bound to another TCP socket [4]. Additionally another socket is required to listen for incoming connections at the same time. Since this is not possible with todays Berkeley (BSD) sockets, the option SO_REUSEADDR must be used which allows multiple socket bindings to the same local endpoint [4].

The NAT traversal procedure is as follows. Both NATed hosts use their active TCP session with the rendezvous server to exchange their public and private endpoints (local and global IP:Port) similar to Section 3.3.2. Using the same local TCP ports that both hosts used with the rendezvous server, they keep sending messages to each other's private and public endpoints while listening for incoming messages [4]. Once the outgoing messages succeed and incoming messages can use the created "hole", the connection must be authenticated to make sure that the other host is really the desired one. In the same test environment as before with UDP, TCP hole punching is supported by about 62% of deployed NATs [4].

3.3.4 Autonomous NAT traversal

A profoundly different method proposed by Müller et al. [11] traverses NAT devices without any third party. We assume that there is a host A behind a NAT device and we are aware of its global IP address, e.g. obtained from a peer list. If an external host B (not behind any NAT device) wants to communicate with this host A, only two messages are required:

1. The NATed host A continuously sends ICMP ECHO REQUESTS to an unallocated IP address, such as 1.2.3.4 [11]. The expected incoming ICMP DESINATION UNREACHABLE packets can be ignored.
2. Because the NAT device now routes corresponding reply messages, the external non-NAT host B sends a fake ICMP TTL_EXPIRED message, which the NAT host A listens for. Such a messages doesn't have to have 1.2.3.4 as source address to be transmitted [11]. Therefore B is the source and the port for future messages is added through the payload as a new ICMP

doi: 10.2313/NET-2014-08-1_01

packet. As A is now aware of B's IP global IP address, it can initiate the actual connection.

A more descriptive representation can be seen in Figure 1 with the two messages numbered likewise.

An implementation of this relatively new approach is the *pwnat* tool by Samy Kamkar. It is also implemented by GNUnet as a transport plugin [12]. Unfortunately for sending these ICMP messages, peers need superuser privileges which may not be available. Müller et al. also propose an alternative method using UDP packets instead of ICMP ECHO REQUESTs which works better on some NAT implementations. Through testing this combined traversal method on a large number of NAT implementations it has shown that the success rate quite remarkable averaging at 82% [11]. Unfortunately this approach has a very little success rate if both hosts are behind NAT devices.

Summarizing this Section, none of these traversal techniques work for every NAT implementation and most of them require a third party. Therefore systems typically implement more than one technique for high success rates [11].

4. CURRENT APPROACHES

There are several approaches which meet the requirements of the previous Section 3 to some extent, but not in its entirety. This shows that this topic still needs research in the future. In this context we distinguish two common bootstrapping methods - using supernodes and existing public overlays. Other notable approaches which do not fit in this Section are discussed in Section 5.

4.1 Supernodes

This model is not a purely decentralized VPN model in which no participant has significantly more features or workload than the others. As implied in Figure 2 these *supernodes* or *superpeers* build a backbone overlay [1, p. 64] and are deployed to deal with the issues of bootstrapping, NAT traversal and routing between the other edge nodes.

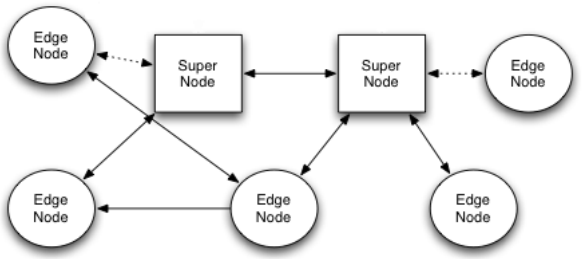


Figure 2: This shows an overlay architecture where supernodes are used for a variety of services, including bootstrapping. Figure source: [13, p. 6]

4.1.1 N2N

N2N is an unstructured layer-two over layer-three P2P VPN [13] released open-source. The software comes with two binaries, the client for edge nodes and the server for supernodes. Technically both can also run on the same physical machine. To create the network, there have to be one or more supernodes which must be directly addressable, that is without a NAT device in between. Usually nodes with high bandwidth and computing power are used as supernodes. Each supernode has to set up communication with one other supernode. The same applies to new edge nodes which require to know the global endpoints of a supernode to join the network. For peer discovery supernodes use layer-two broadcast and they also forward broadcast and multicast packets of the other supernodes to their edge nodes. The result is that each edge node receives these messages and can build up a peer list ({MAC, UDP socket} pair) of the network. As UDP socket we describe the respective endpoint information, that is IP and port.

As previously mentioned there could be middleboxes (see Section 3.3) between supernodes and edge nodes. For the NAT traversal N2N uses UDP hole punching (see Section 3.3.2). Each edge node uses this list to start the so-called peer registration [13] in which it sends out registration requests directly to the other edge nodes. If this UDP hole punching is not successful the specific NAT implementation is probably not supported. If both edge nodes have *symmetric* NATs, the usual NAT traversal is not possible. That is because the global endpoints which are used between an edge node and a supernode cannot be reused with another edge node instead of the supernode. Symmetric NATs would use a different external mapping [13]. In this scenario the supernode works as a relay between the two edge nodes (see Section 3.3.1).

However, N2N is quite inconvenient for end users because bootstrapping and supernode set-up is not automated and has to be deployed by the edge nodes. N2N *1.x* also has some security limitations mentioned on its project site such as the lack of authentication or its vulnerability to replay attacks due to missing nonces in its encryption. A second version with enhanced security extensions is still in developement [13].

Regarding the requirements in Section 3.1 it is questionable if N2N VPNs are scalable with a high amount of users due to the ongoing broadcast and multicast messages.

4.1.2 Tinc

Tinc VPN [19] is more decentralized than N2N because it is organized as a mesh network in which every node relays data for other nodes where it is needed [19]. This may sound similar to supernodes in N2N but every peer has the same functionalities here. The drawback is that this network lacks self-organization and requires explicit specification of which links a node has to create [23]. That means the network has to be organized by hand to be able to deploy tinc's routing protocols [22]. Furthermore tinc doesn't work well if nodes leave. If the missing node was the only one keeping the network together, the network splits in two different ones [22].

4.2 Public Overlays

Another method proposed in [21] is to bootstrap an own private overlay by using existing public overlays with a high availability. This is especially interesting for home users for

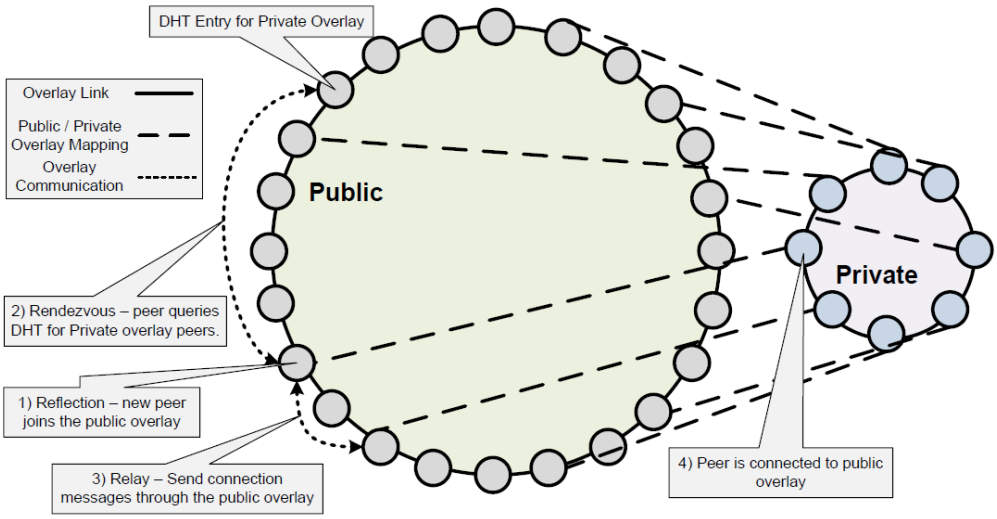


Figure 3: A public overlay is used to bootstrap a private overlay. Figure source: [22, p. 3]

which the configuration and set-up of own rendevouz servers or supernodes may be too difficult or costly.

4.2.1 P2PVPN

The application "P2PVPN" is a decentralized unstructured and open-source VPN [6] and is primarily designed for home users, offering a GUI alternative to N2N and claiming equal funcionality. For bootstrapping it uses the filesharing protocol BitTorrent as a public overlay and entry point. The BitTorrent network is a very popular and large-scale P2P network. The first peer creates the overlay by registering a virtual file on a so-called tracker which works as a rendezvous point now [21]. Following peers who must be aware of its hash value and can obtain the peer list from the tracker. If a tracker goes offline, a different one has to be used. But P2PVPN also offers users to disable the BitTorrent option and bootstrap a purely decentralized network themselves. In this case at least one peer has to be globally routable and must serve as entry point. The first peer can now generate so-called invitations, which are similar to certificates used in the TLS protocol, and only valid for a single peer. They contain all information needed for an external peer to join the network either with the use of a BitTorrent tracker or the direct way with IP and port. The user only has to specify the lifetime of this peer, the other values (signatures, keys) are generated by P2PVPN. These invitations have to be transmitted over a secure channel manually. For peer authentication P2PVPN uses RSA key pairs; inside the network traffic stays encrypted with AES using CBC mode.

Unfortunately NAT traversal is not provided at all and own port forwardings must be configured by hand. Furthermore it has been evaluated that P2PVPN's routing algorithms do not scale well and the software may have problems with large networks [6, p. 61]. That is because update messages between the peers contain their whole routing table.

4.2.2 GroupVPN

In [21] two possible public overlays, XMPP and Brunet, are discussed and extended to the decentralized structured and P2P VPN model called GroupVPN. The *Extensible Messaging and Presence Protocol* (XMPP), also known as Jabber, is a popular XML based messaging protocol [17] which is used in many popular overlays such as GoogleTalk (until 2013) or Facebook Chat [21]. Brunet on the other hand is a freely available DHT (see Section 2) similar to the defunct OpenDHT which was running on the global research network PlanetLab using the Symphony protocol, therefore creating a dedicated bootstrap overlay [21]. A bootstrap overlay consists of one or more bootstrap resources which at least one is publicly available for users with the desire to join. After requesting so called bootstrap nodes from one of these resources it assists in connecting the user with other nodes in the new joined overlay.

The bootstrapping scheme is similar to "P2PVPN" (see Section 4.2.1). For a better understanding Figure 3 is quite helpful: First a node connects with the public overlay and exchanges public endpoint information. This is done using the STUN protocol (compare Section 3.3.2). The next step is to look up the entry of the desired private network in the public networks DHT. Now another peer, which is in both networks, is used for joining the private network.

5. OTHER WORK

This Section mentions approaches of bootstrapping that were not being implemented in P2P software. Despite this, they deal with the bootstrapping problem in a different way as it could be seen in the previous Sections.

In [2] the authors describe an approach of *Local Random Access Probing* where a peer first joins a "bootstrap P2P network" to bootstrap into the actual desired network by name lookup. Of course the problem is just shifted to bootstrapping into this "bootstrap P2P network". That is done by a variant of the existing *Random Access Probing* where a

doi: 10.2313/NET-2014-08-1_01

peer sends messages to random IP addresses until it finds a node which is already part of the bootstrap P2P network. As a difference to that only address ranges around the current IP address of the user are probed. The idea is that private users with Internet access via dialup networks (DSL, cable) would more likely use P2P applications and can profit from an increased communication speed [2]. As an example the eDonkey filesharing network, usually port 4662, is tested in the German IP address space, considering only dial-up networks since the probability for them running such a protocol is higher than others. As a result bootstrapping is successful after around 600 probes which averages in an overall time of 20 seconds. This approach is promising and fulfills most of the requirements of Section 3. The drawback is of course that such a large "bootstrap P2P network" such as eDonkey has to exist in the first place. Furthermore TCP SYN port scan is used as a probing mechanism which requires that the network has a unique port that every peer uses for the protocol.

Another approach uses the *Internet Relay Chat* (IRC) [14], an open, decentralized network of chat servers with high availability [9]. IRC is interesting because once connected to one server of an IRC network, one could communicate with any other client in the same network. Knoll et al. [9] provide a decentralized bootstrapping mechanism for IRC:

1. Connect: Choose an arbitrary IRC server of a specified network and join the chat room, a so-called channel, as a new user with a chosen unique nick name.
2. Discover: If bootstrapping peers (e.g. nick name format "bs_peer") are available in the channel, proceed with the request. If not, this peer becomes a new bootstrapping peer, meaning that a new overlay is created here.
3. Request: The peer requests a peer list and one of the bootstrapping peers responds with a "bootstrapper" list. This list contains contact information of the actual bootstrapping peers. This way the ones that are in the IRC channel are not necessarily the same as the ones that are directly requested outside the channel.
4. Overlay join: With the given list the channel can be left and the peers can be contacted directly. Because these could be hidden behind a firewall or NAT device, a NAT traversal mechanism based on the STUN protocol using supernodes is used [20].

Their approach does not focus on the security and privacy aspects but on the practicality, e.g. an attacker could masquerade as one of the bootstrapping peers and send own lists to new peers.

A second approach by Knoll et al. [8] uses *DDNS*, a dynamic variant of the *Domain Name System*. Dynamic DNS providers offer clients to to map their IP addresses to a (sub)domain name via a password-protected web interface. The idea is that one bootstrapping peer is always mapped to this domain. Clients outside the network can request a peer list using the domain name which is the only thing they have to know about the network. If the current bootstrapping peer leaves, one of the other peers replaces it with its own IP address. The evaluation shows that this approach works surprisingly well with an increasing peer number and with low workload on the DDNS provider (IP address changes). The drawback of this approach is that there always has to be one peer providing the only entry point for the network and thus getting more traffic than the other peers. Furthermore the password for the web interface has to be some kind of well-known which would become a problem if a malicious peer changes this password in the web interface.

6. CONCLUSION

This seminar paper observed the bootstrapping problem and possible solutions. When evaluating existing implementations, not one could fulfill every requirement specified in Section 3.1, although in many cases this was not because it is not technically viable.

In terms of NAT traversal no purely decentralized mechanism existed that worked with all NAT implementations. Nevertheless high success rates can be achieved when using a combination of them, although a third party is often indispensable. Because of the migration to the Internet Protocol version 6 the problem of being globally accessible via end-to-end can simplify middlebox traversal in a significant way. Because of that research is still needed in this area.

But since P2P is becoming more important every day, I cherish great expectations for the developement of decentralized secure peer-to-peer network applications.

7. REFERENCES

[1] J. Buford, H. Yu, and E. K. Lua. *P2P Networking and Applications*. Morgan Kaufmann, 2008.

[2] M. Conrad and H.-J. Hof. A generic, self-organizing, and distributed bootstrap service for peer-to-peer networks. *Self-Organizing Systems*, pages 59–72, 2007.

[3] C. Cramer, K. Kutzner, and T. Fuhrmann. Bootstrapping locality-aware P2P networks. In *Networks, 2004.(ICON 2004). Proceedings. 12th IEEE International Conference on*, volume 1, pages 357–361. IEEE, 2004.

[4] B. Ford, P. Srisuresh, and D. Kegel. Peer-to-peer communication across network address translators. In *USENIX Annual Technical Conference, General Track*, pages 179–192, 2005.

[5] C. GauthierDickey and C. Grothoff. Bootstrapping of peer-to-peer networks. In *Applications and the Internet, 2008. SAINT 2008. International Symposium on*, pages 205–208. IEEE, 2008.

[6] W. Ginolas. Aufbau eines virtuellen privaten Netzes mit Peer-to-Peer Technologie. Master's thesis, Fachhochschule Wendel, `http://p2pvpn.org/thesis.pdf`, 2009.

[7] P. Karbhari, M. Ammar, A. Dhamdhere, H. Raj, G. F. Riley, and E. Zegura. Bootstrapping in Gnutella: A measurement study. *Passive and active network measurement*, pages 22–32, 2004.

[8] M. Knoll, A. Wacker, G. Schiele, and T. Weis. Bootstrapping in peer-to-peer systems. In *Parallel and Distributed Systems, 2008. ICPADS'08. 14th IEEE International Conference on*, pages 271–278. IEEE,

2008.

[9] M. Knoll, M. Helling, A. Wacker, S. Holzapfel, and T. Weis. Bootstrapping peer-to-peer systems using IRC. In *Enabling Technologies: Infrastructures for Collaborative Enterprises, 2009. WETICE'09. 18th IEEE International Workshops on*, pages 122–127. IEEE, 2009.

[10] R. Mahy, P. Matthews, and J. Rosenberg. Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN). Technical report, RFC 5766 (Proposed Standard), 2010.

[11] A. Müller, N. Evans, C. Grothoff, and S. Kamkar. Autonomous NAT traversal. In *10th IEEE International Conference on Peer-to-Peer Computing (IEEE P2P 2010)*, IEEE, 2010.

[12] A. Müller. *Analysis and Control of Middleboxes in the Internet*. Techn. Univ. München, Lehrstuhl Netzarchitekturen und Netzdienste, 2013.

[13] n2n: a Layer Two Peer-to-Peer VPN. `http://luca.ntop.org/n2n.pdf`. `http://www.ntop.org/products/n2n`.

[14] J. Oikarinen and D. Reed. RFC 1459: Internet relay chat protocol. 1993.

[15] OpenVPN. `http://openvpn.net/`.

[16] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing. Session traversal utilities for NAT (STUN). Technical report, RFC 5389 (Proposed Standard), 2008.

[17] P. Saint-Andre. RFC 3920: Extensible messaging and presence protocol (XMPP). *Instant Messaging and Presence, IETFproposed Standard*, 2004.

[18] X. Shen *Handbook of peer-to-peer networking*. Springer, New York, 2010.

[19] I. Timmermans and G. Sliepen *tinc Manual*. `http://tinc-vpn.org/documentation-1.1/tinc.pdf`.

[20] A. Wacker, G. Schiele, S. Holzapfel, and T. Weis. A NAT traversal mechanism for peer-to-peer networks. In *Peer-to-Peer Computing*, pages 81–83, 2008.

[21] D. I. Wolinsky, P. St Juste, P. O. Boykin, and R. Figueiredo. Addressing the P2P bootstrap problem for small overlay networks. In *10th IEEE International Conference on Peer-to-Peer Computing (IEEE P2P 2010)*, IEEE, 2010.

[22] D. I. Wolinsky, K. Lee, P. O. Boykin, and R. Figueiredo. On the design of autonomic, decentralized VPNs. In *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2010 6th International Conference on*, pages 1–10. IEEE, 2010.

[23] D. I. Wolinsky, L. Abraham, K. Lee, Y. Liu, J. Xu, P. O. Boykin, and R. Figueiredo. On the design and implementation of structured P2P VPNs. *arXiv preprint arXiv:1001.2575*, 2010.

doi: 10.2313/NET-2014-08-1_01

Netzwerkexperimente mit NEPI

Tobias Betz
Betreuer: Matthias Wachs
Seminar Future Internet SS2014
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: betz@in.tum.de

KURZFASSUNG

Um das Design, die Implementierung und das Verhalten von Netzwerkprotokollen und Applikationen zu analysieren, greifen Forscher und Entwickler oft auf die Hilfe von Experimenten zurück. Eine Möglichkeit dafür ist die Verwendung von Testbeds. Die verschiedenen Testbeds sind heterogen, was eine Anpassung der Definition eines Experiments bei Benutzung eines neuen Testbed erfordert. Ein einzelnes Netzwerkexperiment ist des weiteren in mehrere Schritte des Deployments, der Steuerung und Analyse aufgeteilt, die für jede Testumgebung anders sind. Dadurch ergibt sich eine hohe Komplexität in der genauen Definition von Netzwerkexperimenten und deren Wiederholbarkeit wird gefährdet.
Dieses Paper gibt eine Einführung in NEPI, einem Framework für Netzwerkexperimente, das die Probleme der Definition von Experimenten und der Vielfalt der zu beachtenden Schritte eines Experiments zu lösen versucht. Hierzu werden die grundlegenden Konzepte von Netzwerkexperimenten betrachtet, wie NEPI diese realisiert, und wie Experimente definiert und ausgeführt werden. Außerdem werden die von NEPI unterstützten Testbeds vorgestellt.

1. EINLEITUNG

Netzwerkexperimente geben Forschern und Entwicklern die Möglichkeit Protokolle und Anwendungen in einer sinnvollen Art zu evaluieren. Durch sie kann das Verhalten analysiert und Probleme bei der Verwendung gefunden werden.
Ein Netzwerkexperiment kann beispielsweise die Skalierbarkeit einer Anwendung nachweisen, oder Fehlverhalten in Umgebungen mit verlustbehafteter Kommunikation aufzeigen. Die Experimente dienen dadurch als Instrument, um neue Ideen zu verwirklichen und vorhandene Konzepte zu validieren.
Es stehen verschiedene Arten von Experimentierumgebungen zur Verfügung. Experimente können auf realen Geräten mit der tatsächlichen Software durchgeführt werden, oder von einem Simulator oder Emulator virtualisiert werden.
Simulatoren verwenden statt der tatsächlichen Software nur Modelle derselben. Dadurch erreichen Simulatoren nicht den gleichen Realitätsgrad wie Emulatoren oder reale Testbeds, weil beispielsweise das benutzte Softwaremodell fehlerhaft sein kann. Andererseits bietet eine Simulation exakte Wiederholbarkeit und gute Kontrolle des Experiments, auch aufgrund der fehlenden Echtzeit-Notwendigkeit.
Die Verwendung von realen Geräten schränkt die Skalierbarkeit von Experimenten stark ein, weil die Verwendung von mehr oder komplexeren Geräten automatisch mit höheren Nutzungskosten verbunden ist. Das Deployment, was der Vorbereitung des Testbeds zur Durchführung des Experiments entspricht, ist bei Simulationen einfacher zu handhaben als bei realen Testbeds.
Emulatoren arbeiten in Echtzeit und instrumentieren die echte Software. Emulatoren stellen eine Art Mischlösung aus Simulatoren und realen Testbeds dar. Sie bieten wegen der Instrumentierung gute Skalierbarkeit, und mehr Realitätsbezug als Simulatoren, sind aber bezüglich Kontrolle und Wiederholbarkeit nicht auf dem gleichen Niveau wie ein Simulator. [1]
In Tabelle 1 werden die verschiedenen Arten von Testumgebungen anhand der genannten Faktoren noch einmal dargestellt.

	Realismus	Kontrolle	Deployment	Wiederholbarkeit
Testbed	+	-	-	-
Emulation	○	○	○	○
Simulation	-	+	+	+

Tabelle 1: Vergleich von physikalischer Umgebung, Emulation und Simulation [2]

Eine Testumgebung kann als reales Testbed oder als Virtualisierung vorhanden sein, wobei für virtuelle Testumgebungen die Technik der Simulation oder Emulation angewendet werden kann. Des weiteren haben konkrete Testumgebungen auch noch andere wichtige Unterschiede und Eigenschaften. So können sich Testumgebungen beispielsweise dadurch unterscheiden, wie viele Geräte für das Experiment zur Verfügung stehen, oder benötigt werden. Das muss aber auch mit den Anforderungen des Experiments verglichen werden.
Außerdem liegt die Zielsetzung von verschiedenen Testbeds in unterschiedlichen Arten von Experimenten. Bei einer Testumgebung deren Hauptgebiet beispielsweise Drahtlose Kommunikationstechnologien sind, macht ein Wireless-Experiment am meisten Sinn. Die möglichen Testumgebungen für ein Experiment sind also sehr heterogen und unterscheiden sich in zentralen Aspekten, wie der inneren hierarchischen Struktur und der damit einhergehenden Art des Zugriffs.
Auch Zugriff auf einzelne Geräte der Testbeds erfolgt unterschiedlich. Ein einfacher Linuxrechner als Teilnehmer einer Testumgebung kann beispielsweise durch das Protokoll SSH gesteuert werden, aber ein verteiltes Netzwerk aus Sensorknoten verlangt nach anderen Arten des Zugriffs.
Der Lebenszyklus eines Netzwerkexperiments läuft in mehreren Schritten ab. Zuerst muss es geplant und definiert werden, um eine einheitliche Darstellung des Experiments vor-

doi: 10.2313/NET-2014-08-1_02

weisen zu können. Hier kann es bereits sein, dass die Eigenheiten der Testbeds miteinbezogen werden müssen. Nach der Definition kann das Experiment ausgeführt werden. Dieser Lebenszyklus kann wieder in Teilschritte unterteilt werden. Zunächst muss der Zugriff auf die teilnehmenden Geräte des Testbeds und deren Konfiguration erfolgen, was dem Deployment entspricht. Anschließend kann das Experiment gestartet und kontrolliert werden. Zum Abschluss werden die Ergebnisse des Netzwerkexperiments evaluiert und in hinsichtlich seiner Fragestellung untersucht. Diese Ergebnisse können als Planungsgrundlage für nachfolgende Experimente dienen.
Durch die erwähnten Unterschiede in den verschiedenen Testbeds, die teilweise sehr umfangreich sind und durch die Anzahl an Schritten, die in einem Experiment notwendig sind, ist die Definition und der gesamte Ablauf eines Netzwerkexperiments für Forscher und Entwickler sehr unübersichtlich.
Es ist sehr aufwendig, bei einer Änderung der Experimentdefinition jeden dieser Aspekte per Hand anpassen zu müssen. Aus diesem Grund ist ein Tool gefragt, das die Komplexität und den Konfigurationsaufwand, der durch die Unterschiede verschiedener Testbeds entsteht, zu minimieren.
In diesem Paper wird mit NEPI[1] ein Werkzeug vorgestellt, das entwickelt wurde um die genannten Probleme zu umgehen. NEPI steht für Network Experiment Programming Interface, und ist ein Framework, das es ermöglicht, auch komplexe Netzwerkexperimente durch ein einzelnes Skript einheitlich zu steuern. Version 3.0 von NEPI wurde im Dezember 2013 veröffentlicht und ist zum aktuellen Zeitpunkt das neueste Release. Auf dieser Version liegt der Fokus in diesem Paper. [3]
In Kapitel 2 wird der grundlegende Aufbau von NEPI erklärt. Anschließend in Kapitel 3 wird der typische Ablauf eines Experiments mit NEPI kurz vorgestellt. In Kapitel 4 wird erläutert, wie ein Experiment in NEPI definiert wird. Danach wird in Kapitel 5 gezeigt, wie die Durchführung von Experimenten funktioniert. In Kapitel 6 werden die von NEPI unterstützten Testbeds und Umgebungen vorgestellt, bevor in Kapitel 8 das Paper noch zusammengefasst wird.

2. STRUKTUR VON NEPI

Wie bereits in der Kapitel 1 erwähnt, ist NEPI ein Framework für Netzwerkexperimente. Es wurde von Planete Team bei INRIA, einer französischen Forschungseinrichtung, entwickelt und ist mittlerweile bei Version 3.0 angekommen. Das nächste Release ist für April/Mai 2014 geplant.
NEPI versucht die Schwierigkeiten, die durch die Unterschiedlichkeiten verschiedener Testumgebungen entstehen, auszugleichen, indem es als einzelnes Werkzeug alle Aspekte zur Durchführung eines Netzwerkexperiments abdeckt. Dadurch ist es möglich, die Konfiguration für Experimente an einem zentralen Punkt zu regeln. NEPI ermöglicht es also, ein Experiment mittels einer einzigen Experimentdefinition, die alle beteiligten Komponenten und Abläufe enthält, festzulegen. Zu den Aufgaben, die NEPI erfüllt, gehört das Deployment, die Kontrolle, die Ausführung und die Ergebnissammlung für ein Netzwerkexperiment. NEPI bietet einen generischen Ansatz für die Ausführung von Experimenten auf verschiedenen Testbeds. Um unterschiedliche Testbeds zu unterstützen, verwendet NEPI ein modulares Prinzip, was die Einbindung noch nicht unterstützter Testbeds ermöglicht. Das zentrale Steuerelement von NEPI heißt *Experiment Controller*. Der Experiment Controller (EC) steuert die *Resource Manager* (RM), die eine gemeinsame Schnittstelle teilen. Ein Resource Manager übernimmt die Steuerung eines Testbed und abstrahiert dessen Funktionen für den Experiment Controller durch Verwendung eines Interface.
In Abbildung 1 ist die Modularität von NEPIs Resource Managern veranschaulicht. Die beiden Resource Manager verwenden hier mit SSH bzw. XMPP unterschiedliche Protokolle zur Steuerung ihrer Testumgebungen.

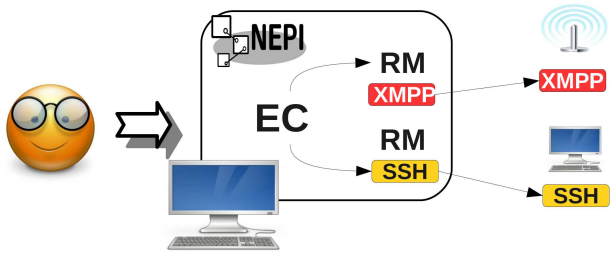


Abbildung 1: Modularität von NEPI [4]

In früheren Versionen von NEPI (v2.0) konnte die Experimentdefinition als "Boxes and Connectors"-Diagramm erfolgen. Dabei werden die beteiligten Geräte und Anwendungen als Boxen dargestellt und ihre topologische Zusammengehörigkeit als Verbindungen zwischen den Boxen. Dieses Diagramm wurde mit einem NEPI-Design-Frontend (NEF) erstellt und konnte direkt verwendet werden. In der aktuellen Version 3.0 ist dieses Feature nicht mehr vorhanden, aber als Veranschaulichung des Experimentdesigns ist das "Boxes and Connectors"-Diagramm trotzdem hilfreich.
NEPI ist als Pythonbibliothek realisiert und ein Benutzer muss nur ein einziges Pythonskript erstellen, das den Ablauf eines gesamten Experiments abdeckt. In diesem Skript wird die Definition des Experiments beschrieben, was bedeutet, dass alle benötigten Ressourcen erzeugt, deren Eigenschaften gesetzt und Abhängigkeiten beschrieben werden. Anschließend kann das Experiment gestartet und Ergebnisse von den teilnehmenden Ressourcen geladen werden. [3]

3. ABLAUF EINES EXPERIMENTS MIT NEPI

Will ein Anwender nun NEPI verwenden, beginnt er mit der Definition des Experiments. Dazu gehört die Beschreibung aller beteiligten Geräte und Abläufe. Diese Definition geschieht in einem Python-Skript, das zur Ausführung des Experiments gestartet wird.
Während der Ausführung kann der Anwender das Experiment kontrollieren, indem er mit der Python-Bibliothek interagiert. Wenn das Netzwerkexperiment beendet ist, ist er in der Lage die gewonnenen Resultate von dem Knoten herunterzuladen, zu analysieren und untersuchen.

[1] `http://nepi.inria.fr/Main/WebHome`

doi: 10.2313/NET-2014-08-1_02

4. EXPERIMENTDEFINITION MIT NEPI

Experimente werden in NEPI mit einem "Boxes and Connectors"-Diagramm definiert und dargestellt. Die Boxen stellen einzelne Ressourcen des Experiments dar, die miteinander verbunden sind.

4.1 Ressourcen

Als *Ressource* kann beispielsweise ein Netzwerkhost, oder eine Netzwerkkarte gesehen werden. Aber auch eine Anwendung, die auf einem Host läuft, eine virtuelle Maschine, oder ein Teil eines Sensornetzes kann eine Ressource repräsentieren. Der Typ einer Ressource ist dann `LinuxHost`, `Interface`, etc.
Jede Ressource hat bestimmte Attribute. Ein Linuxhost hat als Attribute beispielsweise seinen Hostnamen, seine IP-Adresse und den SSH-Port. Für eine Linux-Applikation ist vermerkt, auf welchem Host sie läuft, welche Software dort vorhanden sein muss, und wie der genaue Befehl lautet, der ausgeführt werden soll.
Zur Ergebnissammlung besitzt jede Ressource so genannte *Traces* (engl. Spuren). Bei einer Applikation kann das beispielsweise die Standardausgabe (stdout) und die Fehlerausgabe (stderr) sein, die während des Experiments erzeugt werden.
Die Traces können am Ende des Experiments oder schon während seiner Ausführung vom Benutzer heruntergeladen werden. Sie dienen zur Fehlererkennung, Fehlerbehebung und allgemein zum Debugging. In Abbildung 2 ist die beispielhafte Darstellung einer Linux-Applikations-Ressource gezeigt.

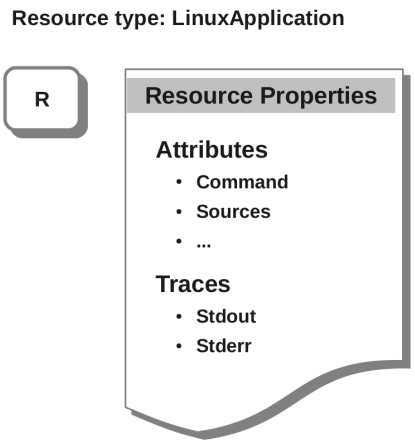


Abbildung 2: Darstellung einer Ressource [3]

Um einen strukturierten Lebenszyklus für Ressourcen zu gewährleisten, hat jede Ressource auch einen Zustand. Dieser Zustand kann beispielsweise angeben, dass die Ressource betriebsbereit ist, noch gefunden werden muss, oder ein Fehler entstanden ist.
Eine Besonderheit von NEPI ist, dass die Zustände der Ressourcen nicht zwingend miteinander synchronisiert sein müssen, d.h. es ist beispielsweise nicht unbedingt nötig, dass eine Ressource zu Experimentbeginn vorhanden ist. [3]
In Abbildung 3 sind die einzelnen Zustände der Ressourcen dargestellt. Der initiale Zustand für alle Ressourcen ist `NEW`, gefolgt von `DISCOVERED`, wenn ein passendes Gerät vom Resource Manager gefunden wurde. Anschließend versucht NEPI die Ressource für den Benutzer verfügbar zu machen, was beispielsweise per SSH-Protokoll passieren kann, und der Zustand wird auf `PROVISIONED` gesetzt. Wenn die Ressource dann konfiguriert wurde, entspricht das dem Zustand `READY`. Eine Ressource kommt in den Zustand `STARTED`, wenn sie erfolgreich gestartet wurde und in den Zustand `STOPPED`, wenn die Ressource für das Experiment nicht mehr benötigt wird.
Wenn der Zustand einer Ressource `FAILED` ist, wurde ein Fehler erkannt, beispielsweise ist die Ressource nicht mehr erreichbar. Der finale Zustand für alle Ressourcen ist `RELEASED`.

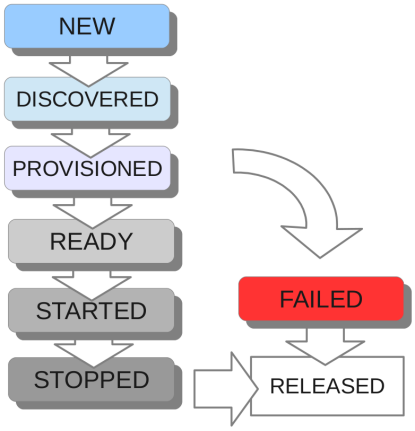


Abbildung 3: Verschiedene Zustände die eine Ressource durchläuft [3]

4.2 Topologie der Ressourcen

Ressourcen hängen oft topologisch voneinander ab. Eine Applikation kann beispielsweise nicht laufen, wenn der dazugehörige Rechner nicht gestartet ist. Das Diagramm, in dem die Ressourcen beinhaltet sind, ist ein topologischer Graph. Darin sind die Ressourcen als Boxen gezeichnet und einige der Ressourcenattribute werden als Verbindungen zwischen den Boxen dargestellt. Eine Linux-Node-Ressource kann als Attribut eine Applikation haben, die auf ihm laufen soll, und die entsprechende Applikations-Ressource hat als Attribut den Linux-Node.
In einem solchen symmetrischen Fall können die Attribute als Verbindungen zwischen Boxen dargestellt werden. Anders ist es bei Attributen wie dem Hostnamen, die nicht als Verbindung zu einer anderen Ressource gesehen werden kann. Diese Attribute sind im "Boxes and Connectors"-Graph nicht zu sehen. Ein weiterer wichtiger Aspekt, der in diesem Graph nicht sichtbar gemacht wird, ist der genaue Ablauf des Experiments.
[3]
In Abbildung 4 wird ein Beispiel eines solchen Graphen gezeigt. Die Detailgenauigkeit ist relativ hoch, weil in diesem Beispiel die beiden Nodes A und B nicht abstrahierend direkt miteinander verbunden sind.
Ein Teilausschnitt des Graphen der die Applikation S und den Host A enthält, ist in Listing 1 als Pythoncode verdeutlicht.

doi: 10.2313/NET-2014-08-1_02

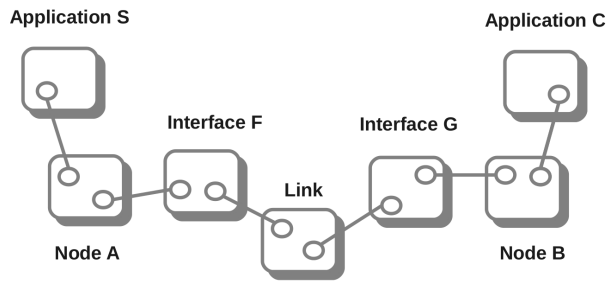


Abbildung 4: Topologischer Graph eines beispielhaften NEPI-Experiments [3]

```
from nepi.execution.ec import
    ExperimentController

ec = ExperimentController(exp_id="my_experiment
    ")
...
app_S = ec.register_resource("LinuxApplication"
    )
ec.set(app_s, "command", my_command)
node_A = ec.register_resource("LinuxNode")
ec.set(node_A, "hostname", my_hostname)
ec.set(node_A, "username", my_username)
...
ec.register_connection(app_S, node_A)
```

Listing 1: Code als Teil eines NEPI-Skripts, der einen Teilausschnitt von Abbildung 4 darstellt

4.3 Abhängigkeiten

Um den ereignisbedingten Ablauf des Experiments aufzuzeigen ist neben dem "Boxes and Connectors"-Diagramm noch ein zusätzlicher Abhängigkeitsgraph nötig. Dieser Graph gibt an, welche Bedingungen erfüllt sein müssen, bevor bestimmte Aktionen ausgeführt werden sollen. Die möglichen Aktionen sind das Deployment für eine Ressource und das Starten und Stoppen derselben. Dabei spielt der in Kapitel 4.1 genannte Status der Ressourcen eine entscheidende Rolle.
In Abbildung 5 ist ein solcher Abhängigkeitsgraph dargestellt. Applikation C soll gestartet werden, nachdem Applikation S gestartet wurde.
In Listing 2 ist die gleiche Abhängigkeit gezeigt, wie sie im Pythonskript aussieht.
[3]

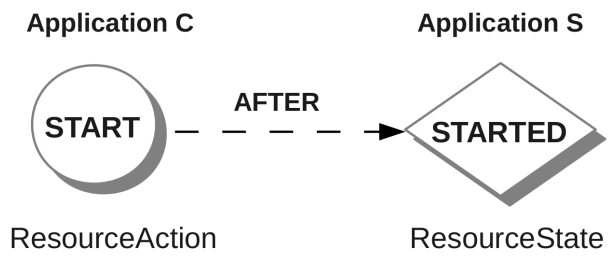


Abbildung 5: Abhängigkeitsgraph zweier Ressourcen [3]

```
from nepi.execution.resource import
    ResourceAction, ResourceState

ec.register_connection(app_C, ResourceAction.
    START, app_S, ResourceState.STARTED)
```

Listing 2: Pythoncode für NEPI, um Abhängigkeiten zu erzeugen

5. DURCHFÜHRUNG VON EXPERIMENTEN IN NEPI

Allein die Beschreibung eines Experiments ist noch nicht ausreichend für eine Durchführung. Für jede Testumgebung muss genau definiert sein, wie Software auf die beteiligten Ressourcen kommt, wie deren Steuerung erfolgt und wie die Testergebnisse gesammelt werden.
Ein Sensorknoten hat aber im Gegensatz zu einem Linux Host keinen SSH-Server, der einen einfachen Login zur Steuerung der Ressource ermöglicht. Deshalb muss für jede Testumgebung ein passender Resource Manager (RM) vorhanden sein der die Steuerung einer spezifischen Ressource übernimmt. Eine Besonderheit von NEPI ist, dass der Benutzer selbst einen Resource Manager für eine neue Testumgebung schreiben kann, der nur die gemeinsame Schnittstelle teilen muss.
Der User interagiert nur mit dem Experiment Controller (EC), der die Schnittstellenfunktionen der Resource Manager nutzt. [4]
In Abbildung 6 ist diese Abstraktion verdeutlicht. Links oben ist der User, der nur den Experiment Controller anspricht, welcher wiederum die Resource Manager über die gemeinsame Schnittstelle (in der Abbildung als "RM API" gekennzeichnet) steuert. Hier ist die Modularität von NEPI deutlich zu sehen. Jeder Resource Manager steuert seine eigenen Ressourcen und ein neuer RM kann problemlos hinzugefügt werden, was in Abbildung 6 als "API x" gekennzeichnet ist.

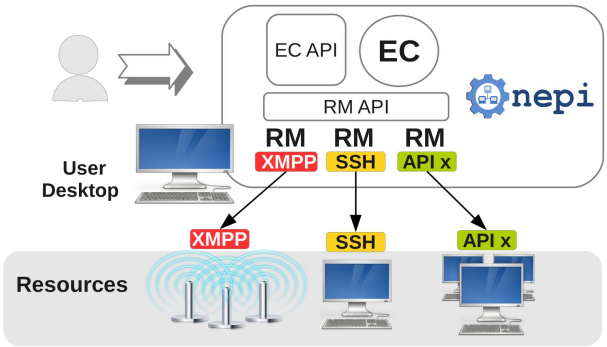


Abbildung 6: Resource Manager kennen die Steuerungsmöglichkeiten ihrer Testumgebungen [3]

Die Ausführung eines Experiments in NEPI kann in drei Phasen aufgeteilt werden, die in Abbildung 7 dargestellt sind. Diese Schritte sollen im Folgenden genauer erläutert werden.

doi: 10.2313/NET-2014-08-1_02

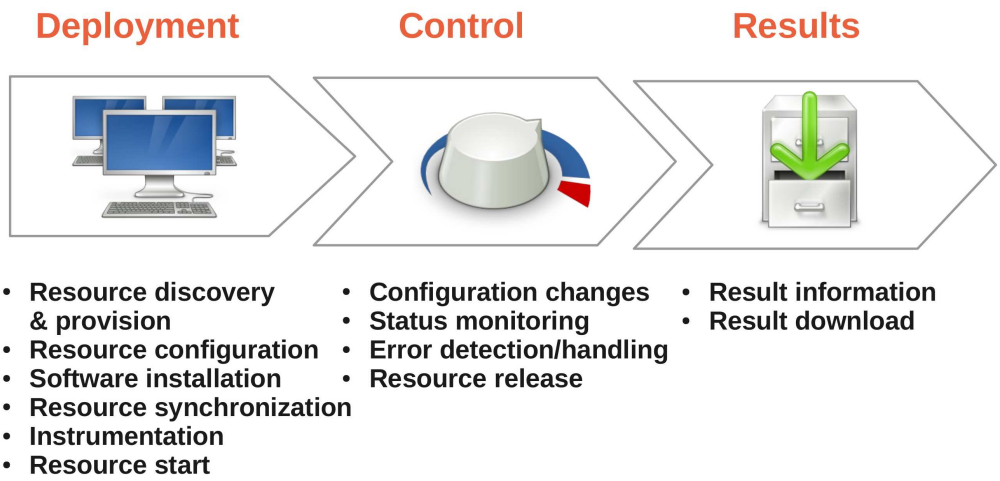


Abbildung 7: Verschiedene Phasen des Lebenszyklus eines Experiments [2]

5.1 Deployment

Das *Deployment* eines Experiments ist das Verteilen der Software auf die Ressourcen und deren Bereitstellung und Initialisierung. Das Deployment wird plattformabhängig von den Resource Managern gehandhabt, d.h. jeder Resource Manager muss wissen, mit welchen Protokollen und Authentifizierungen auf seine Ressourcen zugegriffen werden kann. Der Experiment Controller ruft nur eine `deploy`-Funktion auf, die alle Resource Manager gemeinsam haben.
Um eine Ressource für das Experiment nutzbar machen zu können müssen mehrere Schritte durchgeführt werden. Zunächst muss NEPI sicherstellen, dass die Ressource überhaupt vorhanden und erreichbar ist. Danach kann die Ressource konfiguriert werden, wozu auch die Installation fehlender Software gehört. Sind diese Schritte abgeschlossen versucht NEPI die Ressourcen zu synchronisieren und zu instrumentieren. Anschließend kann die Ressource gestartet werden. [Abbildung 7, links]

5.2 Steuerung

Der Benutzer kann das Experiment nicht nur vor dessen Beginn konfigurieren, sondern auch während der Ausführung noch Änderungen vornehmen. Dazu gehört auch das Hinzufügen von Ressourcen zur Laufzeit, wie bereits in Kapitel 4.1 erwähnt.
Der Benutzer hat die Möglichkeit im laufenden Experiment schon Traces von den Ressourcen herunterzuladen. Sie sind dort gespeichert wo sie erzeugt werden und der zentrale Experiment Controller muss die Traces explizit durch die Resource Manager von den potentiell entfernten Ressourcen anfordern. Diese Traces können zur Statusabfrage und Fehlererkennung genutzt werden. Dazu muss die Ressource im gestarteten Zustand sein. [3]
Ein Experiment wird mit dem Freigeben aller Ressourcen beendet. [Abbildung 7, Mitte]

5.3 Ergebnissammlung

Auch am Ende des Experiments können Traces gesammelt werden. Das ist vor allem für nicht-interaktive Experimente wichtig, die dem Nutzer keine Möglichkeit der Kontrolle während der Laufzeit bieten. Die Traces können von verschiedenen Testumgebungen heruntergeladen werden, was der jeweilige Resource Manager übernimmt.
Je nach benutzten Ressourcentypen gibt es unterschiedliche Arten von Traces. Eine Linux-Applikation hat die Standardausgabe und die Fehlerausgabe, während eine Netzwerkkarte Paketdumps mitschneiden kann. Es ist die Aufgabe des Benutzers aus den vorhandenen Traces die wichtigen auszuwählen, da NEPI über deren Relevanz nicht entscheiden kann. [Abbildung 7, rechts]

6. UNTERSTÜTZTE TESTBEDS

Ein zentraler Aspekt von NEPI ist die Fähigkeit verschiedene Testbeds zu unterstützen. Die Experimente sind nicht beschränkt auf eine einzige Testumgebung, sondern können mit kleinen Anpassungen des Skripts auch auf anderen Testumgebungen ausgeführt werden können.
Im Folgenden werden einige Testbeds vorgestellt, die aktuell, oder zukünftig von NEPI unterstützt werden.

6.1 PlanetLab

PlanetLab ist ein öffentlich zugängliches globales Netzwerk aus tausenden von Hosts, das für Netzwerkexperimente genutzt werden kann. Das Netzwerk ist an das Internet angebunden, und kann somit problemlos von Forschern und Entwicklern benutzt werden, um ihre Experimente unter realen Bedingungen zu testen.
Der Zugriff auf einzelne Nodes erfolgt per SSH mit Schlüsselauthentifizierung. Der Benutzer kann, im Gegensatz zu einem Simulationstestbed, keine beliebigen IP-Adressen an die PlanetLab Hosts vergeben, sondern nur die fest zugeteilten Adressen verwenden. Die Geräte sind nicht für ein einziges Experiment reserviert, es können gleichzeitig mehrere Experimente von unterschiedlichen Entwicklern laufen. Im "Boxes and Connectors"-Diagramm wird ein PlanetLab Host als `Node`-Box verbunden mit einer `Internet`-Box dargestellt. In Abbildung 9 ist dies dreimal zu sehen. [2, 5]

6.2 OMF

Das c*O*ntrol and *M*anagement *F*ramework kurz OMF ist ein Framework für Netzwerktestbeds. Dabei geht es in erster Linie um WiFi-Testbeds. OMF erlaubt im Gegensatz zu PlanetLab nur einen Benutzer pro Host gleichzeitig, ermöglicht es den Entwicklern dadurch aber, den Host beliebig zu

doi: 10.2313/NET-2014-08-1_02

konfigurieren und anzupassen.
Die einzelnen Hosts werden über das XMPP-Protokoll konfiguriert, was für NEPIs Resource Manager eine Rolle spielt. [6]

6.3 ns-3 (zukünftig)

Der Netzwerksimulator ns-3 erlaubt präzise Kontrolle über viele Aspekte der simulierten Hosts und Geräte. Dabei kann die Simulation auf hardwarenahem Level erfolgen. Der Simulator wurde hauptsächlich zu Forschungs- und Bildungszwecken entwickelt, und ist deshalb öffentlich verfügbar. [7]
ns-3 wurde von NEPI v2.0 bereits vollständig unterstützt, in Version 3.0 ist die Einbindung aber noch nicht abgeschlossen. [8]
Die Anbindung an andere teilnehmende Ressourcen des Experiments erfolgt über eine `FileDescriptorNetDevice`-Ressource. Dieses Gerät kann Ethernetframes von einem Linux-Filedescriptor lesen und in Frames übersetzen, die im restlichen Testbed verarbeitet werden können. [1]

6.4 Nemu (zukünftig)

Nemu ist ein Netzwerkemulator, der vom gleichen Team entwickelt wurde, wie NEPI selbst. Der Emulator besteht aus zwei Hauptbestandteilen, netns und netem.
netns wurde bereits in NEPI v2.0 unterstützt, der Support der Kombination von netns und netem durch NEPI v3.0 ist in Arbeit. [8]
netns verwendet virtuelle Maschinen und Interfaces die miteinander verbunden sind. Außerdem wird der Netzwerkstack des Hostbetriebssystems von den virtuellen Maschinen geteilt.
Dadurch hat ein NEPI-Entwickler sehr viele Kontrollmöglichkeiten für das Experiment.[5, 1]

7. KOMBINIERTE VERWENDUNG

Eine Besonderheit von NEPI ist die Möglichkeit, hybride Experimente zu realisieren. Es können mehrere verschiedene Testbeds in einem Netzwerkexperiment kombiniert werden, wodurch beliebige Interaktionen ermöglicht werden.
In den Abbildungen 8 und 9 ist ein Experimentdesign beschrieben, das diese Eigenschaft nutzt. In diesem Experiment werden zwei verschiedene Testbeds benutzt. Rot hinterlegt (Oval, mittig) ist die PlanetLab-Testumgebung, die das Internet simuliert und drei WiFi-Netzwerke, je gelb(links oben), blau (rechts oben) und grün (unten) hinterlegt, die miteinander über das PlanetLab-Netz verbunden sind.
Die drei WiFi-Netzwerke laufen in je einem ns-3 Simulator. Die ns-3-Netzwerke enthalten Hosts, die realistischen Traffic durch VLC-, BitTorrent- oder VoIP-Anwendungen erzeugen.

8. ZUSAMMENFASSUNG

NEPI dient als Werkzeug, um Netzwerkexperimente zu vereinfachen. Die Probleme, die sich durch die Verwendung verschiedener Testbeds und die Komplexität der Ausführungsschritte eines Experiments ergeben, werden durch NEPIs zentralem Management deutlich verringert. Die Wiederholbarkeit der Experimente und die große Anzahl an Einsatzszenarien spricht für NEPI.
Probleme, die das Arbeiten mit NEPI erschweren sind einerseits das Fehlen von NEF, dem grafischen Frontend zum Erstellen der "Boxes and Connectors"-Diagramme sowie die

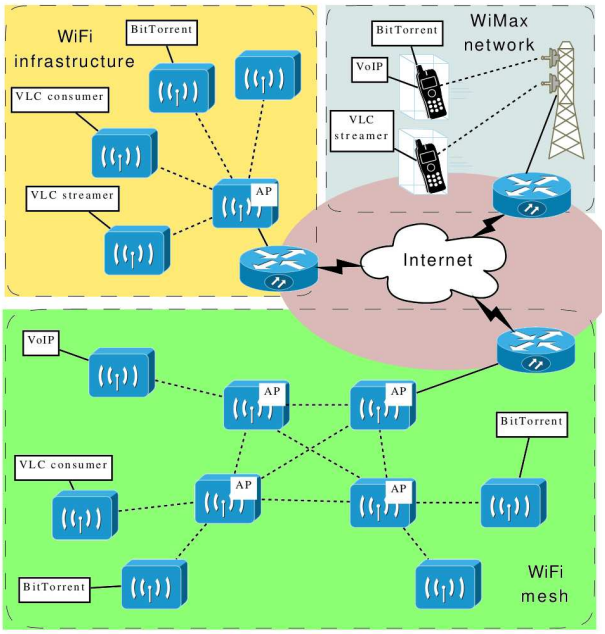


Abbildung 8: Darstellung eines hybriden Experiments [7]

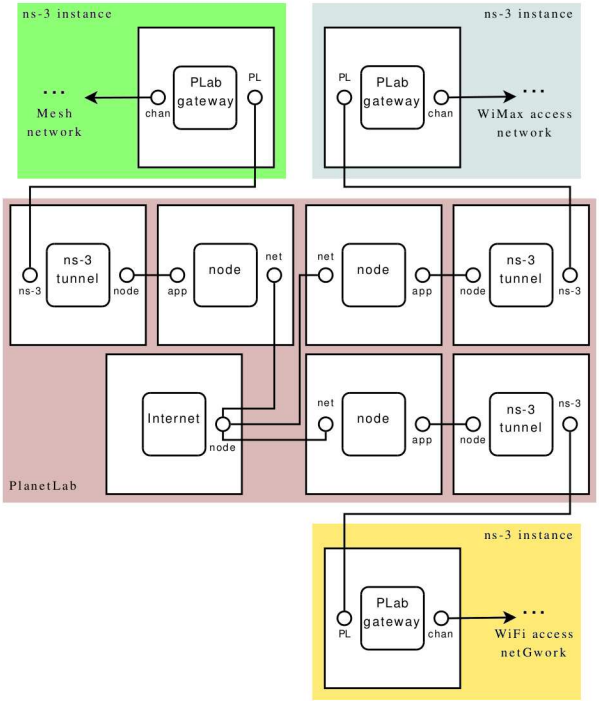


Abbildung 9: Ausschnitt des "Boxes and Connectors"-Diagramms des Experiments aus Abbildung 8 [7]

doi: 10.2313/NET-2014-08-1_02

nicht abgeschlossene Entwicklung von NEPI.
Das zeigt sich darin, dass Publikationen zu Version 3.0 von NEPI nicht vorhanden sind, und die Internetpräsenz des Entwicklungsteams während der Erstellung dieses Papers ständig unter Bearbeitung war. [9] Die einzige Quelle, die die aktuellste Version behandelt ist das ausführliche, aber nicht abgeschlossene User Manual. [3]
Es bleibt zu hoffen, dass diese kleineren Schwachpunkte mit dem neuen Release, geplant für April/Mai 2014, beseitigt werden.

9. LITERATUR

[1] A. Quereilhac, et al.: *NEPI: An integration framework for network experimentation*, In Software, Telecommunications and Computer Networks (SoftCOM), 19th International Conference on. IEEE, 2011

[2] *NEPI hands-on* `http://nepi.inria.fr/pub/Nepi/WebHome/hands_on_nepi.pdf`, [Online, zugegriffen 18. März 2014]

[3] *NEPI v3.0 User Manual* `http://nepi.inria.fr/code/nepi/raw-file/68fabc2fc452/doc/user_manual/user_manual.pdf`, [Online, zugegriffen 17. März 2014]

[4] *Validating ICN implementations* `http://nepi.inria.fr/pub/Nepi/WebHome/NEPI_validating_ICN_implementations.pdf`, [Online, zugegriffer 17. März 2014]

[5] J. P. Rueda: *Testing NEPI usability and features*, Juni 2012

[6] M. Lacage: *Outils d'Expérimentation pour la Recherche en Réseaux*, PhD dissertation, November 2010

[7] M. Lacage, et al.: *NEPI: Using Independent Simulators, Emulators and Testbeds for Easy Experimentation*, In ROADS'09 workshop, Oktober 2009

[8] *Supported platforms* `http://nepi.pl.sophia.inria.fr/Technologies/WebHome`, [Online, zugegriffen 18. März 2014]

[9] *NEPI: Network Experiment Programming Interface* `http://nepi.inria.fr/Main/WebHome`, [Online, zugegriffen 23. März 2014]

doi: 10.2313/NET-2014-08-1_02

Überblick über Botnetz-Erkennungsmethoden

Florian Zipperle
Betreuer: Oliver Gasser
Seminar Future Internet SS2014
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: florian.zipperle@tum.de

KURZFASSUNG

Botnetze sind eine große Sicherheitsbedrohung des Internets, viele gefährliche Angriffe nutzen sie. Bei Botnetzen handelt es sich um von Kriminellen ferngesteuerte Computer, die zu einem Netzwerk zusammengeschlossen sind. Zur Fernsteuerung werden die Computer mit Schadsoftware infiziert. Diese Arbeit gibt eine Einführung und einen Überblick über das Thema Botnetze und wie sie anhand ihrer Architektur und ihrer Kommunikationsinfrastruktur unterschieden werden können. Hauptsächlich gibt es zentrale Infrastrukturen nach dem Client-Server-Model, Peer-to-Peer-Architektur und kombinierte hybride Architekturen. Anschließend wird in der Arbeit eine mögliche Kategorisierung der aktuellen Methoden der Botnetzerkennung gezeigt. Die Verfahren werden in signatur- und verhaltensbasierten Methoden eingeteilt. Durch Netzwerküberwachung wird nach auffälligem Verhalten und bekannten Mustern gesucht, um Bots im Netzwerk zu erkennen. Zu den einzelnen Kategorien werden aktuelle Forschungsergebnisse vorgestellt.

Schlüsselworte

Botnetze, Botnetz-Erkennung, Übersicht

1. EINLEITUNG

Diese Arbeit soll einen Überblick über Botnetze und die Methoden zu ihrer Erkennung geben. Botnetze sind wie Viren und Würmer eine große Sicherheitsbedrohung des Internets, sowie seiner Infrastruktur. Botnetze sind mitunter die gefährlichsten von ihnen [8]. Deshalb ist eine gute und zuverlässige Erkennung notwendig und Botnetz-Erkennung ein aktuelles Forschungsgebiet.

Dabei strukturiert sich diese Arbeit im weiteren Verlauf folgendermaßen: im ersten Kapitel werden Botnetze kurz beschrieben und ihre Verwendungszwecke aufgezeigt, in Kapitel zwei folgt eine genauere Kategorisierung und Beschreibung von Botnetzen. In Kapitel drei wird eine Klassifizierung der verschiedenen Ansätze zur Botnetzerkennung beschrieben. Anschließend in Kapitel vier wird ein Überblick über konkrete Forschungsergebnisse gegeben. In Kapitel fünf folgt noch der Schluss der Arbeit mit einem Ausblick.

1.1 Was sind Botnetze?

Der Name Botnetz setzt sich zusammen aus Bot und Netz, Bot kommt von Roboter. Sie sind Netzwerke bestehend aus Computern, welche von Schadsoftware befallen sind. Durch diese Schadsoftware können vom sogenannten Botmaster das Netzwerk bzw. die einzelnen Computer gesteuert werden. Die Steuerung erfolgt über eine sogenannte Command-and-Control-Infrastruktur (C & C) , welche im Verborgenen abläuft. Ein Computer, welcher sich in einem Botnetz befindet, wird meist ohne Kenntnis des Besitzers missbraucht [18].

1.2 Bedrohung durch Botnetze

Botnetze werden meist für kriminelle Zwecke missbraucht. Da Botnetze mehrere tausend Computer umfassen können, bieten sie eine große Rechenleistung sowie eine große Datenbandbreite im Internet für den Angreifer [9]. Der Angreifer ist im Falle von kriminellen Aktivitäten zusätzlich durch die Schicht der Botnetze geschützt, d.h. für das Opfer ist der Angreifer nicht sichtbar sondern nur ein übernommener Computer des Botnetzes. Als zusätzlichen Schutz vor Strafverfolgung kann der Botmaster den Angriff so planen, dass zwischen dem Befehl und dem tatsächlichen Angriff eine beliebige Zeitspanne herrscht. Dadurch wird es schwieriger die Betreiber des Botnetzes zur Rechenschaft zu ziehen [26].

Einige Bedrohungen und Verwendungszwecke von Botnetzen sind [5] [18]:

- **DDoS:** Als Distributed-Denial-of-Service, kurz DDoS, bezeichnet man eine Attacke gegen einen Server, bei welcher sehr viele Rechner gleichzeitig fehlerhafte bzw. unsinnige Anfragen an einen Server schicken. Durch die Flut der fehlerhaften Pakete kann der Server überlastet werden und gültige Anfragen nicht mehr beantworten. Dadurch kann er seine eigentliche Aufgabe nicht mehr wahrnehmen. Dies stellt insbesondere für eCommerce Anbieter ein Problem dar, da sie dadurch wirtschaftlichen Schaden erleiden können.

- **Spam:** Die übernommenen Computer können dazu missbraucht werden Spam zu versenden. Diese Spam Mails können verwendet werden, um Schadsoftware zu verteilen, welche das Botnetz vergrößern. Durch das Versenden von Spam kann ein Rechner auf eine Blacklist kommen und dadurch keine Mails mehr versenden. Laut einer Schätzung von 2009 erzeugen Botnetze 85 % der über 100 Milliarden Spamnachrichten, die täglich versendet werden [15].

- **Klickbetrug:** Durch Klickbetrug kann mit Botnetzen Geld verdient werden. Dabei richtet der Botnetzbetreiber zuerst ein Konto bei einem Online-Werbesystem wie z.B. Google AdWords ein und stellt anschließend

doi: 10.2313/NET-2014-08-1_03

eine Webseite online, welche diese Werbung anzeigt. Danach steuert er die Bots so, dass sie seine Seite besuchen und die Werbung anklicken. Da dies meist ohne des Wissens des Computerbesitzers im Hintergrund geschieht, entspricht dies nicht dem Ziel der Werbetreibenden und es wird deshalb von Klickbetrug gesprochen. Der Botnetzbetreiber kann diesen Service auch vermieten, sodass andere an der Werbung verdienen.

- **Identitätsdiebstahl:** Durch das Aufzeichnen der Tastatur des befallenen Computer können Passwörter oder Kreditkartendaten gestohlen werden. Diese Daten können für Identitätsdiebstahl missbraucht werden und Onlinekonten können dadurch übernommen werden. Es kann auch gezielt nach gespeicherten Passwörtern und Benutzerdaten auf dem Rechner gesucht werden. Die gestohlenen Informationen können vom Botnetzbetreiber, zur Aufrechterhaltung seines Botnetzes genutzt werden, z.B. um Domains zu reservieren ohne seine eigene Identität preis zu geben.

2. BOTNETZE

Die Botnetze besitzen meist den gleichen Lebenszyklus, unterscheiden sich jedoch oft in ihrer Architektur sowie in ihrer Art der Kommunikation.

2.1 Lebenszyklus

Der Lebenszyklus eines Bots bzw. dessen Schadsoftware innerhalb eines Botnetzes lässt sich grob in fünf Abschnitte einteilen[5]:

- **Infizierung:** Zu Beginn wird ein Computer durch die Botsoftware infiziert. Dies geschieht durch Sicherheitslücken im System oder durch Schadsoftware, die durch den Benutzer des Computers ausgeführt wird.
- **Sammeln:** Falls die Infizierung erfolgreich war und eine Verbindung zum Internet besteht, meldet sich die Schadsoftware über den Command-and-Control-Server beim Botmaster.
- **Sichern und Verstecken:** Die Schadsoftware muss sich im System verstecken, um nicht vom Benutzer entdeckt und entfernt zu werden. Für diesen Zweck wird auch versucht gebräuchliche Antivirensoftware zu deaktivieren bzw. sie zu manipulieren. Außerdem kann versucht werden zusätzliche Schadsoftware über das Internet herunterzuladen.
- **Befehle ausführen:** Der Bot ist einsatzbereit und kann verwendet werden. Er wartet auf Befehle vom Botmaster. Sobald er Befehle über die Command-and-Control-Infrastruktur erhält, versucht er diese auszuführen. Dies können Updates der Botsoftware sein, um die Entdeckung zu erschweren oder um neue Funktionen zu erhalten. Es können auch die in 1.2 beschriebenen Attacken ausgeführt werden.
- **Tod:** Der Tod eines Bots kann mehrere Ursachen haben. Der Bot kann entdeckt worden sein oder das Betriebssystem des Computers wird neu installiert. Es kann auch vorkommen, dass der Botmaster aus Vorsicht die Schadsoftware vom Computer löscht, um nicht erkannt zu werden und keine Spuren für Forscher und Ermittler zu hinterlassen.

2.2 Architektur

Es existieren hauptsächlich zwei verschiedene Architekturen, eine zentralisierte und eine verteilte Struktur. Die zentralisierte Struktur setzt auf das Client-Server-Modell und die verteilte Struktur auf Peer-to-Peer-Protokolle [18].

2.2.1 Zentralisierte Struktur

In der zentralisierten Struktur wird auf das im Internet üblichen Client-Server-Modell zurückgegriffen. Der Bot fungiert als Client und der Botmaster als Server. Der Botmaster betreibt einige wenige zentrale Server, die zur Steuerung und Kommunikation mit den Bots dienen. Da alle Bots mit den zentralen Servern verbunden sind, kann der Botmaster mit allen zeitgleich und schnell kommunizieren. Er erhält direkte Rückmeldungen über den Status und die Größe seines Botnetzes. Die zentralen Server bieten allerdings eine gute Angriffsfläche für Gegenmaßnahmen. Durch das Ausschalten der zentralen Server kann das ganze Botnetz lahmgelegt werden.

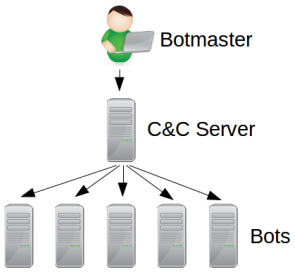


Abbildung 1: Zentralisierte Architektur mit einem Command-and-Control-Server, über welchen der Botmaster sein Botnetz steuert.

Als Beispiel für ein Botnetz mit zentralisierter Architektur kann das Botnetz Rustock genannt werden. Es wurde 2006 das erste Mal wahrgenommen und konnte 2011 mit Hilfe von Reverse Engineering unschädlich gemacht werden [7].

2.2.2 Verteilte Struktur

Um das Botnetz robuster zu gestalten, setzen Botnetzentwickler auf eine verteilte Architektur. Diese basiert meist auf dem Peer-to-Peer-Prinzip. Dabei gibt es nicht einen Server und viele Clients, sondern jeder Bot kann die Rolle eines Servers und eines Bots übernehmen. Die Erkennung, wer der Botmaster ist, fällt dadurch schwerer. Die Befehle, die der Botmaster gibt, werden im Botnetz verteilt. Die Bots tauschen sich dafür aus, der Bot mit dem älteren Befehl verwirft diesen und übernimmt den neuen Befehl. Dadurch, dass die Befehle nicht zeitgleich von einem zentralen Server geholt werden können, sondern sich erst im Netz ausbreiten müssen, kann dies zu einer geringeren Reaktionszeit führen [17][18].

In [25] werden die zwei Peer-to-Peer-Botnetze Strom und Nugache vorgestellt. Die beiden werden einander gegenübergestellt und verglichen. Beide setzten auf eine verteilte Struktur, Nugache besitzt jedoch auch eine zentralisierte Kommunikationsmöglichkeit über IRC. In der Praxis wird diese jedoch kaum verwendet. Beide Botnetze verwenden eine Verschlüsselung für die Kommunikation. Das Strom Bot-

doi: 10.2313/NET-2014-08-1_03

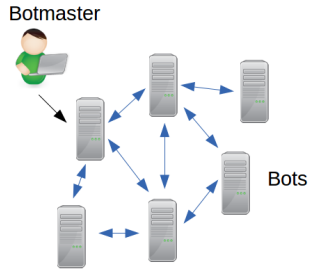


Abbildung 2: Verteilte Architektur, die Bots sind untereinander vernetzt und verteilen die Befehle des Botmasters selbstständig im Netz.

netz setzt für die Kommuniktion auf dein bestehendes Protokoll namens Overnet. Damit ein neuer infizierter Rechner sich dem Botnetz anschließen kann, braucht er einen Kontaktpunkt. Dieser Kontaktpunkt wird durch andere Bots die sich bereits im Botnetz befinden geben. Die IP-Adressen verschiedener Kontaktpunkte sind dabei hart-kodiert im Schadprogramm enthalten. Der neue Bot versucht Verbindungen zu den Kontaktpunkten herzustellen und sich dem Botnetz anzuschließen. Die Liste mit den Kontaktpunkten wird dabei aktualisiert.

2.2.3 Hybride Architektur

Die Vorteile beider Architekturen können in einem hybriden Ansatz vereint werden. Dabei existieren mehrere verteilte Netze mit jeweils einigen wenigen zentralen Servern. Dadurch ist das Netz robust gegen Gegenmaßnahmen und kann trotzdem schnell und übersichtlich gesteuert werden [22].

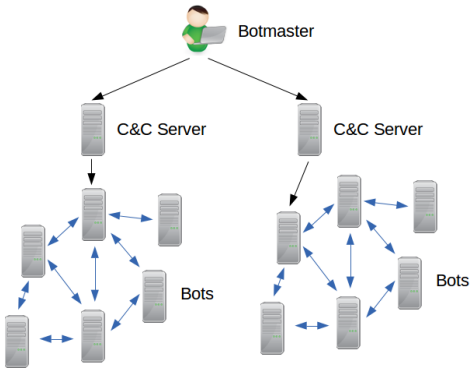


Abbildung 3: Die hybride Architektur stellt eine Kombination dar. Es gibt einige verteilte Netze, welche jeweils über einen eigenen zentralen Command-and-Control-Server verfügen. Dies kombiniert die Vorteile der zentralisierten und verteilten Architektur.

Ein Beispiel für ein konkretes Botnetz, welches auf eine hybride Architektur setzte, nannte sich Waledac. Es setzte einige zentrale Server ein. Diese Server steuerten ein darunterliegendes selbst entwickeltes Peer-to-Peer-Protokoll. Waledac wurde im April 2008 das erste Mal entdeckt und wurde 2010 im Februar erfolgreich ausgeschaltet [7].

2.3 Kommunikation

Zur Kommunikation mit dem Botnetz und zur Realisierung der Command-and-Control-Struktur können verschiedene Infrastrukturen und Methoden verwendet werden. Die ersten Botnetze verwendeten meist Internet-Relay-Chat (IRC) Protokolle zur Kommunikation. IRC ist für große Chaträume ausgelegt und es existieren einige quell-offene Implementierungen. Dies eignet sich, um als Botmaster schnell und einfach mit dem Botnetz zu kommunizieren [10]. Es existieren auch weiterhin viele Botnetze, die auf IRC setzen. Viele der komplexen und gefährlichen Botnetze setzen jedoch verschlüsselte Kommunikation und keine zentralen Server ein [7][29].

Ein anderer Ansatz zur Kommunikation ist es normalen HTTP-Netzwerkverkehr zu verwenden. Dabei können ganz normal per GET und POST Daten ausgetauscht werden. Die Kommunikation lässt sich aber auch verstecken, indem z.B. über eine öffentliche Kommentarfunktionen eines sozialen Netzwerks die Daten getauscht werden [12]. Befehle lassen sich ebenfalls in Bildern oder anderen Dateien verstecken. Diese können über soziale Netzwerke verteilt werden. Dadurch wird die Erkennung erschwert, da der Netzwerkverkehr kaum von normalem unterscheidbar ist [30].

Peer-to-Peer-Netze verwenden teilweise bestehende Peer-to-Peer-Netze, die für Filesharing ausgelegt sind. So wird Beispielsweise das Framework WASTE, welches ein verschlüsseltes Peer-to-Peer-Protokoll zur privaten Kommunikation implementiert, verwendet [12]. Durch die Verwendung von Peer-to-Peer-Protokollen kann weitestgehend auf einen zentralen Server verzichtet werden und es kann eine verteilte Architektur verwendet werden. Die Übersicht für den Botmaster über sein Botnetz, welche zum Beispiel bei IRC gegeben ist fehlt, eben weil der zentrale Punkt fehlt.

3. ERKENNUNGSVERFAHREN

Die verschiedenen Methoden zur Erkennung von Botnetzen lassen sich wie in diesem Kapitel dargestellt einteilen. Dabei werden verschiedene Methoden und Ansätze je nach Architektur und Kommunikationsinfrastruktur verwendet. In Abbildung 4 ist eine mögliche Kategorisierung der Erkennungsverfahren gegeben. Die einzelnen Kategorien werden in den folgenden Unterkapiteln näher betrachtet.

Erkennungsmethoden lassen sich in aktive und passive Erkennungsmethoden unterteilen. Aktive Verfahren bieten dem Botmaster die Möglichkeit zu erfahren, dass sein Botnetz untersucht wird. Dadurch kann er sein Botnetz ändern und es durch bessere Methoden absichern und schützen. Passive Verfahren hingegen sind für den Botmaster nicht erkennbar [3].

3.1 Honeynet

Honeynets sind Netzwerke, die ein oder mehrere Honeypots enthalten. Als Honeypot wird in diesem Zusammenhang ein Computer bezeichnet, der keine produktive Funktion hat außer Schadsoftware anzulocken. Um mit Schadsoftware infiziert zu werden, wird oft auf veraltete mit Sicherheitslücken behaftete Software verwendet. Jeder Netzwerkverkehr, der durch ein Honeypot verursacht wird, deutet auf Schadsoftware hin. Eingehender Netzwerkverkehr kann z.B.

doi: 10.2313/NET-2014-08-1_03

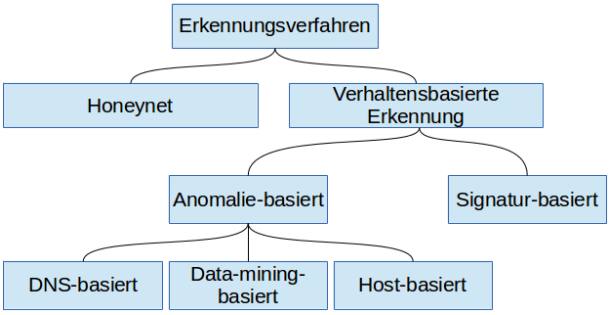


Abbildung 4: Einteilung der verschiedenen Erkennungsmethoden für Botnetze [1].

ein Portscan sein oder sonstige Attacken. Entsteht ausgehender Netzwerkverkehr, bedeutet dies dass der Computer von Schadsoftware befallen ist [13]. Aus diesen Daten lassen sich Schlüsse über die Art des Botnetzes ziehen und wie es kommuniziert. Es können neue Schwachstellen entdeckt werden, welche es ermöglichen in das System einzudringen. Außerdem können die Vorgehensweise und die verwendeten Werkzeuge des Angreifers analysiert werden. Honeynets werden von Forschern eingesetzt oder auch von Internet Service Providern, um zu erkennen, ob in ihrem Netz Bots eines Botnetzes vorhanden sind [19].

3.2 Verhaltensbasierte Erkennung

Honeynets sind nützlich, um bekannte Botnetze zu verstehen und zu analysieren. Falls ein Computer infiziert ist, aber nicht versucht andere Computer ebenfalls zu infizieren, bleibt er für ein Honeynet unentdeckt. Für die Erkennung solcher Bots sind die Ansätze von verhaltensbasierten Verfahren besser geeignet [10]. Unter verhaltensbasierter Erkennung werden Methoden zusammengefasst, welche versuchen das besondere Verhalten eines Bots zu erkennen. Dies geschieht meist durch Analyse und Beobachtung des Netzwerkverkehrs, das ein Computernetzwerk generiert. Dies kann z.B. durch einen Internet-Service-Provider geschehen.

3.2.1 Anomalie-basierte Erkennung

Das Verhalten von Computern wird auf Anomalien untersucht. Dies können hohes Netzwerkverkehrsvolumen, Verwendung von unüblichen Ports oder sonstiges unübliches Systemverhalten sein. Diese Ansätze können nochmals feiner gegliedert werden [1]: **DNS-basierte Erkennung**
Bei DNS-basierten Erkennungsverfahren wird vor allem der durch DNS erzeugte Netzwerkverkehr untersucht. Als Domain-Name-System (DNS) wird der Dienst bezeichnet, welcher Domains in IP-Adressen auflöst. Dies wird von Botnetzen verwendet, um mit dem Command-and-Control-Server eine Verbindung aufzubauen. Durch die Zuhilfenahme von DNS müssen die IP-Adressen der Server nicht im Code des Bots stehen, sondern können dynamisch aufgelöst werden. Dies ist nützlich für einen Serverumzug.

Ein Grund weshalb DNS-basierte Methoden zur Erkennung verwendet werden, ist dass der durch DNS erzeugte Netzwerkverkehr im Verhältnis sehr klein ist. Dadurch fällt das Analysieren und Auswerten der Daten an einem großen Netzwerkknoten einfacher. Es wird also weniger Rechenleistung benötigt, wodurch unter Umständen der Netzwerkverkehr in Echtzeit durchsucht werden kann[23].

Domains können relativ einfach gesperrt werden, falls bekannt ist, dass sich dahinter ein Botnetzbetreiber befindet. Um die daraus resultierende Gefahr für Botnetze zu vermeiden, werden sogenannte Domainname-Generation-Algorithmen (DGA) verwendet, um dynamisch neue Domains bzw. Subdomains zu erzeugen. Möchte der Bot mit dem Comand-and-Control-Server kommunizieren, generiert er solange neue Domains mit Hilfe eines DGA bis er einen gültigen Server findet. Dies liefert einen Angriffspunkt gegen Botnetzbetreiber [23].

Data-Mining-basierte Erkennung
Data-Mining bedeutet aus gegebenen Daten neues Wissen zu sammeln. Bei Data-Mining-basierten Erkennungsmethoden werden Verfahren aus dem Bereich Data-Mining angewandt, um den überwachten Netzwerkverkehr zu klassifizieren und zu gruppieren. Gruppierung wird bei vielen Erkennungsverfahren eingesetzt. Dies ist oft der erste Schritt, bei welchem versucht wird der von einem Computer verursachte Netzwerkverkehr in zwei Klassen einzuteilen. Es wird versucht der Netzwerkverkehr, der von Bots generiert wird, vom normalen Netzwerkverkehr zu unterscheiden und die infizierten Computer in einem Netzwerk zu erkennen. Dadurch ist es möglich große Datenmengen auf kleiner Datenmengen zu reduzieren, wodurch weniger Rechenleistung erforderlich ist. Dies ermöglicht unter Umständen Echtzeituntersuchgen oder komplexere Algorithmen. Es werden Algorithmen für maschinelles Lernen angewandt, diese können zum Beispiel zur Erzeugung von Signaturen verwendet werden [1] [3].

Host-basierte Erkennung
Bei dieser Herangehensweise zur Erkennung von Botnetzen wird nicht das Netzwerk überwacht und untersucht, sondern die Suche findet auf den Computern statt. Ist ein Computer von Schadsoftware befallen und der Bot aktiviert, lässt sich dies oft durch ungewöhnliche Systemaufrufe erkennen. Die Schadsoftware versucht sich im System zu verstecken und z.B. Antivirenscanner zu deaktivieren. Ein Bot versucht auch Verbindungen in das Internet aufzubauen [14].

3.2.2 Signaturbasierte Erkennung

Signaturbasierte Verfahren sind ähnlich zu Anomalie-basierten Methoden [28]. Sie verwenden vorhandenes Wissen über Botnetze, dieses Wissen kann mit Hilfe von Honeynets gesammelt werden. Die daraus resultierende Signatur bzw. das typische Verhalten wird in einfache Regeln übertragen. Ein Computer gilt als verdächtig, wenn er diese Regeln erfüllt [4]. Das Prinzip ist ähnlich wie bei einem Virenscanner auf einem Computer [6]. Da vorhandenes Wissen verwendet werden muss, um die Regeln aufzustellen, eignen sich reine signaturbasierte Methoden schlecht, um unbekannte Botnetze zu erkennen.

3.3 Hybride Verfahren

Die konkrete Einteilung in diese genannten Kategorien ist für reale Erkennungsverfahren schwierig. Da die meisten Verfahren Methoden aus den verschiedenen Bereichen verwenden. Die Kategorisierung gibt aber trotzdem einen groben Überblick welche Verfahren eingesetzt werden. Diese Verfahren welche Kombinationen enthalten werden als hy-

doi: 10.2313/NET-2014-08-1_03

bride Verfahren bezeichnet werden. Da durch die Kombination der einzelnen Methoden die Stärken der verschiedenen Ansätze vereint und die Schwächen kompensiert werden können. Diese hybriden Verfahren haben deshalb meist bessere Erkennungsraten oder bessere Laufzeiten [1].

In [2] wird ein hybrider Ansatz präsentiert, der Host-basierte und Data-mining-basierte Methoden vereint, um Peer-to-Peer-Botnetze zu finden.

4. FORSCHUNGSERGEBNISSE

Hier werden einige konkrete Forschungsergebnisse präsentiert. Zunächst werden verschiedene Kriterien gezeigt, die zur Bewertung der Güte einer Erkennungsmethode verwendet werden können.

4.1 Bewertung von Erkennungsverfahren

Eine hohe Erkennungsrate ist wünschenswert, es werden jedoch weitere Ansprüche an Erkennungsverfahren gestellt. Sie sollten eine kleine falsch-positiv Erkennung haben. Eine falsch-positiv Erkennung bedeutet in diesem Fall, dass ein Computer zu unrecht beschuldigt wird ein Bot zu sein. Dies kann negative Folgen für einen unschuldigen Internetnutzer haben, beispielsweise können alle sein Mails zu unrecht als Spam verworfen werden.

Es ist wünschenswert, dass sie unabhängig von der Struktur und der Art der Kommunikation sind, um vielfältig einsetzbar zu sein. Das heißt auch, dass sie verschlüsselte Kommunikation erkennen sollen.

Methoden, die auf eine Netzwerkverkehrsanalyse setzen, sollten mit großen Datenmengen umgehen können und echtzeitfähig sein.

Die Erkennung von Bots sollte idealerweise geschehen bevor diese größeren Schaden anrichten können.

	Signatur-basiert	DNS-basiert	Mining-basiert	Honeynet
Unbekannte Botnetze	Nein	Ja	Ja	Ja
Protokoll und Struktur unabhängig	Nein	Nein	Ja	Ja
Verschlüsselte Kommunikation	Nein	Ja	Ja	Ja

Tabelle 1: Eine grobe Übersicht für welche Zwecke, welche Art von Erkennungsverfahren prinzipiell geeignet ist [20].

In Tabelle 1 wird eine grobe Übersicht gegeben für welche Botnetze welche Erkennungsverfahren eher geeignet sind und für welche nicht. Dies ist im allgemeinen schwierig zu sagen, aber es können Tendenzen aufgezeigt werden. So sind zum Beispiel signatur-basierte Ansätze schlecht geeignet um neue noch unbekannte Botnetze zu erkennen. Da die typische Signatur erst erarbeitet werden muss. Ebenso ist es bei DNS-basierten Verfahren, diese setzen voraus, dass DNS verwendet wird und sind somit nicht Protokoll und Struktur unabhängig.

4.2 Konkrete Erkennungsverfahren

DNS-basierte Erkennung

Botnetzbetreiber verwenden häufig dynamische DNS-Dienste (DDNS), dies bietet einen Ansatz Bots zu erkennen. Computer machen sich verdächtig, wenn sie häufig DDNS-Domains auflösen. DDNS-Dienste verwenden meist kurze TTL-Werte (Time-to-live), der Trend geht jedoch dazu, auch für normale DNS-Einträge kleine TTL-Werte zu wählen. Dies hat zur Folge, dass viele falsche Computer verdächtigt werden [27].

Ein anderer Ansatz nutzt DNS-basierte Black-hole Lists (DNSBL). DNSBL enthalten IP-Adressen von Spammern bzw. Computern, die Schadsoftware verteilen. Diese Listen können über DNS Abfragen genutzt werden z.B. folgendermaßen: <IP-Adresse>.spamhaus.org. Wird diese Anfrage korrekt aufgelöst, ist die IP-Adresse in der Datenbank aufgelistet [24]. In [21] wird beschrieben wie dies zur Erkennung genutzt werden kann. Falls Botnetzbetreiber bzw. die Bots selbst prüfen, ob sie in den Datenbanken gelistet sind, können sie entdeckt werden. Durch Überwachung dieser Abfragen können verdächtige Computer gefunden werden. Dies hat zudem den Vorteil, dass Bots meist vor Attacken ihren Status überprüfen und dadurch eine Erkennung vor der Attacke selbst möglich ist.

Wie in 3.2.1 beschrieben bietet ein eventuell eingesetzter Domainname-Generation-Algorithmus eine Möglichkeit Botnetze zu erkennen. So hat zum Beispiel Conficker.C 50000 Domains pro Tag generiert. Um zukünftige Domains zu sperren, muss der DGA des Bots durch Reverse Engineering untersucht werden. Ein Update der Bots kann die ganze Arbeit zunichte machen, falls ein neuer Algorithmus eingesetzt wird. Die Erkennung solcher Bots basiert darauf, dass die Domains sich signifikant von normalen Domains unterscheiden. Dabei können Methoden aus dem Bereich Data-Mining für die Gruppierung verwendet werden. Die alphanumerische Verteilung unterscheidet sich meist stark zwischen maschinell generierten Domains und normalen Domains. Die Bots erzeugen viele gleiche DNS-Abfragen und viele davon können nicht aufgelöst werden, da die Domains nicht registriert sind, dies macht einen Computer verdächtig [23].

Data-Mining-basierte Erkennung

BotHunter ist eine konkrete Anwendung zur Botnetzerkennung, das Verfahren welches eingesetzt wird beruht auf Korrelation [11]. Die Idee dahinter ist es, typische Muster für die verschiedenen Abschnitte des Lebenszyklus zu erkennen. In [29] wird auf BotHunter aufbauend ein Ansatz geliefert, der auch mit verschlüsselter Kommunikation zurecht kommt.

In [31] wird ein Data-Mining-basiertes Verfahren vorgestellt. Der Ansatz geht auch von Netzwerkverkehrsüberwachung eines Netzwerks aus. Dabei wird der Netzwerkverkehr in Zeitscheiben eingeteilt und die einzelnen Zeitscheiben unabhängig von den anderen analysiert. Der Inhalt der Pakete wird ignoriert, d.h. es kann auch verschlüsselte Kom-

doi: 10.2313/NET-2014-08-1_03

munikation erkannt werden. Zur Analyse werden Algorithmen für maschinelles Lernen eingesetzt. In diesem Fall wird auf Entscheidungsbäume gesetzt. Es gibt zwei Phasen, einmal die Lernphase, dabei wird das System mit normalen Netzwerkverkehr und Netzwerkverkehr der Botnetze enthält, trainiert. Die zweite Phase bildet die Erkennungsphase, dabei werden die gelernten Muster gesucht. Das bedeutet der Entscheidungsbaum wird abgearbeitet, um am Ende zu entscheiden, ob ein Bot im Netzwerk ist.

Host-basierte Erkennung
In [2] wird ein kombinierter Ansatz aus Host- und Netzwerküberwachung beschrieben, um speziell Peer-to-Peer-Botnetze zu erkennen. Auf dem einzelnen Computer wird dabei das Dateisystem des Betriebssystems überwacht, um eine Einnistung in das System zu erkennen. Außerdem wird überwacht, ob sich ein Programm in die Autostartliste einträgt und der Netzwerkverkehr wird mitgeloggt.

Signaturbasierte Erkennung
In [4] wird ein konkretes signaturbasiertes System vorgestellt. Dabei wurde der ausgehende Netzwerkverkehr eines Netzwerks mit Hilfe einer Signaturdatenbank analysiert. Bekannte Botnetze konnten in einem Versuch in realen Netzwerken erkannt werden.

Ein weiterer signaturbasierter Ansatz wird in [16] beschrieben. Dabei werden einfache Regeln aufgestellt, die eine klare Syntax haben. Sie bestehen aus Header, Inhalt und Botnetzname. Der Inhalt besteht aus dem speziellen Verhalten des Botnetzes z.B. das Auflösen bestimmter URLs, das Scannen anderer Computer im Netz mittels TCP SYN Paketen oder das Herunterladen von Dateien mit bestimmten binärem Inhalt. In einem Experiment wurde der Ansatz mit bekannten Bots getestet. Diese Bots setzten auf verschiedene Architekturen z.B. Peer-to-Peer. Auch Botnetze mit zentralem Command-and-Control-Server, die über IRC oder HTTP kommunizieren, wurden getestet. Es konnten Erkennungsraten mit durchschnittlich über 80 % erreicht werden. Mit den gleichen Regeln konnten auch unbekannte Botnetze mit über 50 % erkannt werden, ohne unschuldige Computer zu melden, d.h. ohne falsch positive Verdächtigung.

5. SCHLUSS

Botnetze stellen eine immer größer werdende Bedrohung für das Internet dar, durch DDoS-Attacken, Spam-Mails oder Datendiebstahl. Deshalb ist es sehr wichtig sie zu erkennen und auszuschalten. Es gibt zahlreiche Ansätze diese Ziele zu erreichen und durch ihre Kombination werden immer bessere Erkennungsraten erreicht. Diese verschiedenen Ansätze zu vergleichen ist jedoch relativ schwierig, da sie mit unterschiedlichen Voraussetzungen getestet wurden. Zum Teil wurden reale Netzwerke überwacht zum Beispiel in Zusammenarbeit mit Internet-Service-Providern oder Universitätsnetzwerken. Es wurden jedoch auch im Labor bekannte Botnetze aufgesetzt und dabei die Erkennungsrate ermittelt. Das Feld Botnetzerkennung ist und bleibt auch weiterhin ein spannendes Forschungsfeld, da sich die Botnetzbetreiber den neuen Möglichkeiten anpassen und dadurch versuchen die Erkennung ihrer Botnetze zu verhindern.

Zukünftige Geldeinnahmequellen für Botnetzbetreiber könnten Bitcoins sein. Die virtuelle Währung wird durch aufwenden von Rechenleistung erzeugt. Hier könnten Botnetzbetreiber anonym die Rechenleistung ihrer Bots zu Geld machen. Für die Zukunft könnten mobile Smartphones zu beliebten Angriffszielen werden. Die Rechenleistung steigt immer weiter und häufig läuft veraltete Software auf den Geräten. Zusätzlich sind viele Smartphones ständig mit dem Internet verbunden und im Betrieb. Dadurch kann ein Botnetzbetreiber zu jeder Zeit auf seine Bots zugreifen.

6. LITERATUR

[1] R. S. Abdullah, M. F. Abdollah, M. Noh, Z. Azri, M. Z. Mas' ud, S. R. Selamat, and R. Yusof. Revealing the criterion on botnet detection technique. *International Journal of Computer Science Issues (IJCSI)*, 10(2), 2013.

[2] R. S. Abdullah, M. F. Abdollah, Z. A. M. Noh, M. Z. Mas' ud, S. Sahib, and R. Yusof. Preliminary study of host and network-based analysis on p2p botnet detection. In *Technology, Informatics, Management, Engineering, and Environment (TIME-E), 2013 International Conference on*, pages 105–109. IEEE, 2013.

[3] E. Alparslan, A. Karahoca, and D. Karahoca. Botnet detection: Enhancing analysis by using data mining techniques, 2012.

[4] S. Behal, A. S. Brar, and K. Kumar. Signature-based botnet detection and prevention. *http://www. rimtengg. com/iscet/proceedings/pdfs/advcom p/148. pdf*, 2010.

[5] D. Bhatt, N. Garg, and R. Rawat. A survey on botnet detection. In *Proceedings of National Conference on Trends in Signal Processing & Communication (TSPC '13)*, volume 12, page 14th, 2013.

[6] N. Davis. Botnet detection using correlated anomalies. *Technical University of Denmark Informatics and Mathematical Modelling*, 2012.

[7] D. Dittrich. So you want to take over a botnet. In *Proceedings of the 5th USENIX conference on Large-Scale Exploits and Emergent Threats*, pages 6–6. USENIX Association, 2012.

[8] M. Eslahi, R. Salleh, and N. B. Anuar. Bots and botnets: An overview of characteristics, detection and challenges. In *Control System, Computing and Engineering (ICCSCE), 2012 IEEE International Conference on*, pages 349–354. IEEE, 2012.

[9] M. A. R. J. Z. Fabian and M. A. Terzis. My botnet is bigger than yours (maybe, better than yours): why size estimates remain challenging. In *Proceedings of the 1st USENIX Workshop on Hot Topics in Understanding Botnets, Cambridge, USA*, 2007.

[10] M. Feily, A. Shahrestani, and S. Ramadass. A survey of botnet and botnet detection. In *Emerging Security Information, Systems and Technologies, 2009. SECURWARE'09. Third International Conference on*, pages 268–273. IEEE, 2009.

[11] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee. Bothunter: Detecting malware infection through ids-driven dialog correlation. In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, page 12. USENIX Association, 2007.

[12] N. Hachem, Y. Ben Mustapha, G. G. Granadillo, and

H. Debar. Botnets: lifecycle and taxonomy. In *Network and Information Systems Security (SAR-SSI), 2011 Conference on*, pages 1–8. IEEE, 2011.

[13] honeynet Project. Know your enemy: Honeynets. `http://old.honeynet.org/papers/honeynet/`.

[14] O.-R. Jeong, C. Kim, W. Kim, and J. So. Botnets: threats and responses. *International Journal of Web Information Systems*, 7(1):6–17, 2011.

[15] J. P. John, A. Moshchuk, S. D. Gribble, and A. Krishnamurthy. Studying spamming botnets using botlab. In *NSDI*, volume 9, pages 291–306, 2009.

[16] W. M. Li, S. L. Xie, J. Luo, and X. D. Zhu. A detection method for botnet based on behavior features. *Advanced Materials Research*, 765:1512–1517, 2013.

[17] K. Muthumanickam and E. Ilavarasan. P2p botnet detection: Combined host-and network-level analysis. In *Computing Communication & Networking Technologies (ICCCNT), 2012 Third International Conference on*, pages 1–5. IEEE, 2012.

[18] D. Plohmann, E. Gerhards-Padilla, and F. Leder. Botnets: Detection, measurement, disinfection & defence. *European Network and Information Security Agency (ENISA)*, 2011.

[19] N. Raghava, D. Sahgal, and S. Chandna. Classification of botnet detection based on botnet architechture. In *Communication Systems and Network Technologies (CSNT), 2012 International Conference on*, pages 569–572. IEEE, 2012.

[20] N. Raghava, D. Sahgal, and S. Chandna. Classification of botnet detection based on botnet architechture. In *Communication Systems and Network Technologies (CSNT), 2012 International Conference on*, pages 569–572. IEEE, 2012.

[21] A. Ramachandran, N. Feamster, and D. Dagon. Revealing botnet membership using dnsbl counter-intelligence. *Proc. 2nd USENIX Steps to Reducing Unwanted Traffic on the Internet*, pages 49–54, 2006.

[22] R. A. Rodríguez-Gómez, G. Maciá-Fernández, and P. García-Teodoro. Survey and taxonomy of botnet research through life-cycle. *ACM Computing Surveys (CSUR)*, 45(4):45, 2013.

[23] R. Sharifnya and M. Abadi. A novel reputation system to detect dga-based botnets. In *Computer and Knowledge Engineering (ICCKE), 2013 3th International eConference on*, pages 417–423. IEEE, 2013.

[24] T. Sochor and R. Farana. Improving efficiency of e-mail communication via spam elimination using blacklisting. In *Telecommunications Forum (TELFOR), 2013 21st*, pages 924–927. IEEE, 2013.

[25] S. Stover, D. Dittrich, J. Hernandez, and S. Dietrich. Analysis of the storm and nugache trojans: P2p is here. *USENIX; login*, 32(6):18–27, 2007.

[26] W. T. Strayer, D. Lapsely, R. Walsh, and C. Livadas. Botnet detection based on network behavior. In *Botnet Detection*, pages 1–24. Springer, 2008.

[27] R. Villamarín-Salomón and J. C. Brustoloni. Identifying botnets using anomaly detection techniques applied to dns traffic. In *Consumer Communications and Networking Conference, 2008. CCNC 2008. 5th IEEE*, pages 476–481. IEEE, 2008.

[28] K. Wang, C.-Y. Huang, L.-Y. Tsai, and Y.-D. Lin. Behavior-based botnet detection in parallel. *Security and Communication Networks*, 2013.

[29] H. Zhang, C. Papadopoulos, and D. Massey. Detecting encrypted botnet traffic. In *Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on*, pages 163–168. IEEE, 2013.

[30] Z. Zhang, X. Cui, and C. Liu. Web2bot: Botnet in web 2.0 era.

[31] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, and D. Garant. Botnet detection based on traffic behavior analysis and flow intervals. *Computers & Security*, 39:2–16, 2013.

doi: 10.2313/NET-2014-08-1_03

Botnet Detection with DNS Monitoring

Christopher Will
Advisor: Oliver Gasser
Seminar Future Internet SS2014
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: christopher.will@tum.de

ABSTRACT

Botnets are today the universal tool for malicious activities in the Internet. They can send out spam messages, host fairly redundant malicious webpages, perform DDoS attacks and do much more. Of course, researchers have therefore been trying to effectively find and shut down botnets as quickly as possible. The Domain Name System has become an important part of such botnets, for both the botmaster and the defender. It enables botmasters to either hide their content servers via fast-flux, but also offers a good possibility to communicate with the Command and Control server in the background with the help of Domain Generation Algorithms. This paper gives an overview on recent developments in the field of detecting botnets with the help of the Domain Name System and evaluates the different solutions in terms of required input, practicability, efficiency and privacy.

Keywords

Botnet Detection, Domain Name System

1. INTRODUCTION

Within the past years, the Internet has become a vital part of the lives of billions of people. Today, there are many services that are crucial for everyday life and an outage of servers and services can cause great damage, both monetarily and in terms of reputation. Distributed Denial of Service (DDoS) attacks initiated by botnets are often responsible for such an outage disabling a server to answer to any legitimate traffic because of too many queries by the attacker. But botnets are capable of other malicious activities as well: Their infrastructure can be used to redundantly host content which can range from spam pages that aim to sell some replicated products over phishing sites to pages distributing malware or sending spam or phishing mails. Many large botnets have been identified or even taken down in the past, like Torpig [16], Conficker [11] or Mega-D [12]. But nevertheless, as the papers presented here suggest, there are still a lot of botnets in the Internet that might not have even been detected yet.

A very important part of the botnet is the communication structure between the bot and the botmaster. One way to maintain the communication and receive instructions is the usage of one or a few so-called Command and Control servers (C&C server). This is especially important for the first time after a new host was infected. The question is how to get the address of that server: If an IP is hardcoded, it can easily be identified and blacklisted or taken down; the whole botnet would be rendered useless. The Domain Name System (DNS) is a much better way to initialize the communication because the mapping from a domain to one or more IPs can be done dynamically. This mechanism, however, is still easy to stop because now the domain name can be blocked, which is easily and with little effort achievable. The solution found by malware developers are Domain Generation Algorithms (DGA). The bot malware is able to pseudo-randomly generate a high number of domain names that are all with very high probability not already registered. The botmaster only registers very few of these domain names. Nevertheless, all generated domain names are sent to the DNS server so that the bot gets to know the current IP of the C&C server. This complicates the countermeasures to be taken significantly: Even if a botnet malware is reverse engineered and the possible domain names are known in advance, it is possible for a botmaster to modify the DGA very fast or use obfuscation techniques to cover the DGA [1]. Moreover, even if both the currently active domain name as well as the current C&C IP are blacklisted, the malware is able to reconnect to the botmaster (using another domain and another IP). This can also not be prevented by just registering all generated domains, because they are simply too many. Therefore, other detection techniques have to be applied to discover botnet communication between the zombies and the C&C server.

In addition to the usage of DGAs, which is also called domain-fluxing, some botnets use the so-called fast-flux technique to host pages redundantly. A botmaster can never be sure about how long each bot is online. Therefore, he uses once again the DNS mapping to rapidly change the returned IP addresses so that, e.g., phishing pages keep staying accessible. By monitoring the DNS replies, it is also possible to find such botnets. However, it is important to note that the Round-Robin DNS method and Content Distribution Networks (CDNs), which are used for legitimate load balancing of large web sites, also periodically change the returned IPs for a DNS query. That legitimate traffic has to be distinguished from the illegitimate bot traffic.

This paper provides the technical background to understand the mechanisms to detect botnets with the help of DNS traffic. Then, it gives examples of frameworks that aim to do that as research has been very active in that area in the past few years. In total, one paper dealing with fast-fluxing as well as four papers presenting different systems to detect DGA-based botnet malware are examined. Additionally, the

doi: 10.2313/NET-2014-08-1_04

approaches are compared and analysed with respect to required input, practicability, efficiency and privacy.

In the following Section, we will give a technical background of modern-day DNS usage as well as the structure of a botnet. Section 3 deals with the detection of botnets employing the fast-flux technique to host content, Section 4 gives an overview on different approaches to detect DGA botnets by scanning DNS traffic. Section 5 concludes with a short summary and final thoughts.

2. TECHNICAL BACKGROUND

In this Section, we describe how modern and large web services as well as botnets use the DNS for load balancing and redundancy. Additionally, a short description of today's botnet structures is given.

2.1 Domain Name System

The main concept of the Domain Name System (DNS) can be found in the RFCs 1034 [13] and 1035 [14]. In a nutshell, the DNS provides the mapping service between a domain name and the corresponding IP. If a client wants to access a web page, it sends a query to its local DNS resolver requesting the IP address of the domain name. If no cached DNS response is found there, the resolver asks its pre-defined DNS server for the IP. If it also finds no cached entry for the name, it initiates a cascade of requests to the required zone servers according to the domain name. Once the IP or IPs are found, they are returned to the client, including a TTL number indicating how long that DNS answer should be cached without another lookup. If no mapping is found, a corresponding message is sent to the client as well.

Modern web services with a lot of traffic rely on different techniques involving the DNS to balance the load and mitigate possible DoS attacks [8]. A simple method to distribute the load is the round-robin DNS (RRDNS) method: The DNS server loops through a list of possible servers and returns a different IP for each different request until the beginning of the list is reached again. However, this approach is only suited for rough load balancing [3].

The second, more sophisticated method is the application of Content Distribution Networks (CDNs) [10]. The content distributor is normally a third company which is only responsible for intelligently providing and distributing content so that the load is balanced. To achieve that, the content servers are distributed around the world and the corresponding authoritative DNS server belongs to the CDN. Once it receives a query, it takes several parameters into account, including the availability of resources of the CDN and the distance between requesting client and the possible servers, calculates the currently best IP match and sends it back to the client [8]. To ensure an ongoing balancing, the TTL is low so that a client has to query the DNS server again quickly. This also means that it is not uncommon that the same client gets different IPs – maybe even different locations around the world – each time it queries a DNS lookup.

2.2 Botnets

Beside the botnets with one or more central C&C servers, there are also P2P-based botnets [4]. However, this paper focuses on the ones with C&C servers as P2P botnets have a different communication structure. After a new machine is infected, it attempts to contact a C&C server to get instructions on what to do next. Only then, the new bot becomes an active part of the botnet. If no connection can be established, the bot can practically not be used and is worthless. Different methods have emerged and maybe even combined to make sure that a connection is established. Just like ordinary web site traffic, one option is exploiting the DNS service to establish a reliable connection with the help of Domain Generation Algorithms (DGAs). Bots using DGAs generate pseudo-randomly domain names and send all of them to the DNS server. Most of these queries will fail as the botmaster only registers very few of these domains to map to the C&C IP. This provides a robust communication technique. Nevertheless, other techniques have also been used already, for example with the help of social networks: Kartaltepe et al. report that e.g., Twitter was used as command structure by using an account that posts Base64-encoded messages for the bots [9]. Figure 1 shows the layout of a sample C&C-based botnet. One part of it (Bot 1) could serve as content hoster, some other part could perform a DDoS attack at some server. In the background, the C&C server tells the bots what to do. Of course, it is also possible that the whole botnet is executing the same order.

The DNS techniques explained in Section 2.1 are employed by legal web services with the aim of load balancing. A botmaster who wants to host some content is mainly interested in availability as he has to expect that each bot could drop out at any time. This is where the so-called fast-flux method comes into play [8]: In such a system, the bots act as proxies redirecting traffic to some central content server. Additionally, the TTL of a DNS query is once again very low and each time a new DNS query is started, a different set of IPs of bots is returned. This means that if one bot fails or is taken down, the whole network is not affected. Moreover, this quick change of IPs makes it harder to blacklist a small set of hosts and adds redundancy to the system.

3. FAST-FLUX DETECTION

Botnets can be found by analysing either DNS queries and responses sent by oneself or by just listening and analysing generated DNS traffic by others, depending on the situation. If a botnet uses the fast-flux system to reliably provide some content with a malicious background, it can be found by active queries. The following Section discusses that detection technique in detail. In that scenario, the DNS traffic between some client attempting to get to the hosted content and the DNS server providing the IPs for the different proxy bots is interesting.

A fast-flux system aims at redundantly providing content that is mostly illegal – e.g., pages selling replicas, phishing pages, malware distributing sites – by exploiting DNS capabilities [8]. In this scenario, the bots act as proxy servers that redirect any client request to a server in the background actually hosting the content. A user is directed to these pages, e.g., by spam mails. As a single bot is much more likely to fail and suddenly be unresponsive to the botmaster, it is important for him that he can quickly react on such an event and provide a client with functional proxies as fast as possible. This is achieved with the fast-flux technique. The

doi: 10.2313/NET-2014-08-1_04

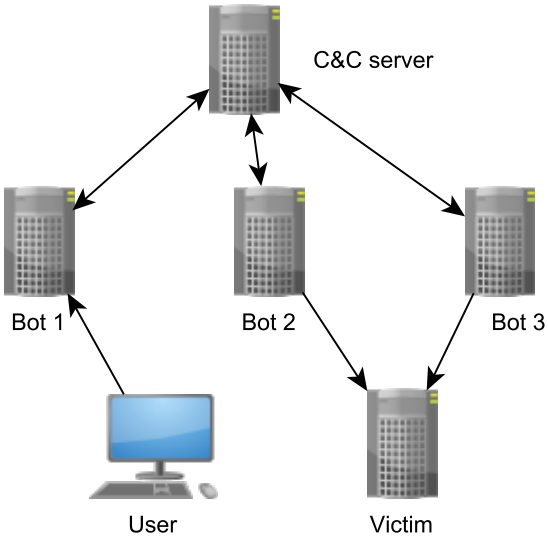


Figure 1: Sample layout of a botnet with a central C&C server in the background.

DNS responses of a domain name with a fast-flux network in the background have a very low TTL (for example, 600 seconds). Additionally, after each new lookup, a new set of IPs different to the one before is returned. This way, it is not a big problem if one or even more proxies fail as there are enough fallback IP addresses.

This technique has similarities with both the round-robin system and CDNs. As fast-fluxing is mainly used for illegal activities and the other ways for legitimate services, it is important to be able to distinguish between these two groups. As the RRDNS method usually employs longer TTLs, a distinction from it can be achieved relatively easily by examining only DNS responses with a low TTL [8].

In general, the distinction between a fast-flux network and a CDN is is more difficult. For the sake of load balancing, the TTL is low there as well. This means that other parameters need to be used as well. Holz et al. suggest to have a closer look at three different values: The number of unique A records (IPs) returned in a series of queries n_A, the number of different Autonomous System Numbers (ASN) n_{ASN} which specify the different network operators and ultimately the different locations of the servers whose IPs are used and the number of nameserver records (NS) in one lookup n_{NS} [8].

In a next step, they tried to calculate a "Flux-Score" for a domain name by defining a function that is a linear combination of the mentioned parameters with some weights w_1 to w_3:

$$f(x) = w_1 \cdot n_A + w_2 \cdot n_{ASN} + w_3 \cdot n_{NS}$$

With the help of a set of benign and fast-flux domains and a 10-fold cross-validation, they came to the following allocation of the weights: $w_1 = 1.32$, $w_2 = 18.54$, $w_3 = 0$. If $f(x) > 142.38$, then a domain is classified as fast-flux, if $f(x) \leq 142.38$, it is considered a benign domain. Using these values and a set of 128 fast-flux domains and 5803 benign domains, 99.98% of all domains were correctly classified.

The allocation of the weights clearly shows that the most distinctive feature of fast-flux networks is that the bots are spread on a large area, possibly around the world. The reason for that is of course that a botmaster cannot choose which machines get infected and takes all bots he can get around the whole world. As CDNs make also use of a lot of different IPs, the number of returned A records is a much weaker indicator of a botnet in the background. Lastly, the amount of different nameserver records plays no role at all and can be completely disregarded. Empirical studies showed then that fast-flux domains can have several thousand different A records and even hundreds of different ASNs [8].

Once a domain using fast-fluxing is found, several steps can be taken to mitigate its harmfulness or even to take down the underlying botnet [8]. A simple method would be to blacklist the affected domains, e.g., by a firewall. By employing a database of fast-flux domains, the authors state that spam filters can be improved as all mails containing domains in the database could be considered spam right away. In order to further analyse the botnet and eventually take it down, a cooperation with the Internet Service Provider (ISP) is required. The bots acting as botnets could be sent an identifiable request while they are watched to identify the content hosting server in the background that could also be the C&C server of the botnet. However, the proxies could use DGAs to complicate the identification and termination of the hosting server in the background. We will look at this method in the next Section.

In conclusion, fast-fluxing botnets can be identified and separated from legitimate DNS load balancing techniques very well and with a low rate of failure. However, as Holz et al. also mention, botmasters might attempt to clone the behavior of a CDN. But as the availability of bots is never guaranteed and as the botmaster will always choose availability to make money over hiding the botnet, botnets will in our opinion never look so similar to CDNs that they cannot be detected anymore (even though an adjustment of the weights of the Flux-Score might be necessary now and then).

As the returned IP addresses for the same host within a longer period of time have to be analysed, the identification requires to actively send DNS queries. This might require resources and could be detected by the botnet. Other systems to identify fast-flux networks have been proposed as well, which are largely based on the parameters presented here [2]. Lastly, it should of course be noted that not all botnets are used to host content, but do something different, like performing a DDoS attack. If that is the case, fast-fluxing detection techniques cannot be applied.

4. DGA DETECTION

In this Section, we present and evaluate several approaches that target identifying the communication between bots and the C&C server while using Domain Generation Algorithms (DGAs) and the Domain Name System (DNS). First, the general structure of the different frameworks is explained,

doi: 10.2313/NET-2014-08-1_04

then, the different approaches are presented.

4.1 General Structure

The initial assumption is always the same: Next to all the benign DNS queries, there might always be traffic from a botnet malware that uses a DGA to get the IP address of the C&C server. This means that the malware creates pseudo-randomly a large amount of domain names. These domains have then of course a certain structure (mostly, they are longer than legitimate domains and consist of a sequence of letters and numbers). All of these domains are then requested at a DNS server. The botmaster registers only a small set of these domains for his C&C servers so that most domain resolutions will fail, only a few will reveal the IP address of the C&C server. This opens up possibilities to detect this rather unusual DNS traffic. The frameworks examined always take similar steps to find that traffic. They are shown in Figure 2 and described in the following.

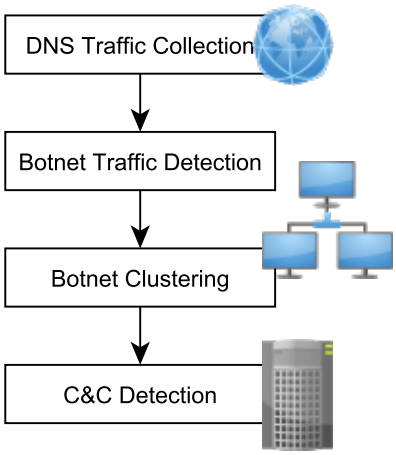


Figure 2: General structure of frameworks detecting DGA-based botnets with DNS monitoring.

DNS Traffic Collection
First of all, the detection framework needs a DNS traffic input. This traffic can be previously collected from a DNS server, but it may also be live data while the system is currently attached to and scanning traffic at a DNS server. It is common to first test the system offline with some recorded input and then evaluating if it is possible to let it run live. This mass collection and analysis of data raises privacy questions, of course. Some frameworks deal with this issue, some others do not. The topic will be mentioned again at each single evaluation.

Botnet Traffic Detection
The next step is the most important one: The identification of the botnet DNS traffic. Naturally, the most part of the DNS queries are benign and completely uninteresting for us. Therefore, some algorithms and/or learning techniques have to be applied to filter out all the benign queries so that only the botnet traffic remains. There are a lot of different methods to identify the botnet queries. The most common one is the exploitation of the fact that botnet malware using a DGA sends out a lot of queries that result in a reply stating that the domain does not exist. As such a reply is relatively uncommon as typos of users etc. are rare compared to all the correct DNS queries, this method poses a very good starting point for further, more detailed filtering.

Botnet Clustering
Normally, a lot of bot traffic of different botnets is found in the previous step. In order to better analyse them, the bots are in some frameworks further grouped according to certain parameters. This provides overview because it is common that different malware uses different DGAs that have different features and can be separated. This way, it is possible to find out which bots might belong to the same botnet and if some bots belong to a previously detected botnet or are part of a yet unknown botnet (or have at least an unknown DGA).

C&C Detection
After having identified the different botnets, the last step is to get to know the IP address of the heart of the botnet, the C&C server. As the bots are known by now, they simply have to be watched to see which of their DGA-generated DNS requests are actually successful.

This is only a rough outline of the different frameworks. There might be more steps involved, like the training of the framework or anonymization, sometimes less. The different approaches are now presented.

4.2 Pleiades

The Pleiades system by Antonakakis et al. is a very sophisticated framework which uses the "non-existing" resolution replies from a DNS server (NXDomains) to detect new classes of DGAs [1]. It consists of two different modules, the *DGA Discovery* module and the *DGA Classification and C&C Detection* module. The first one is responsible for detecting and clustering botnet queries with the help of NXDomains. The latter one is responsible for generating a model that helps finding out whether a correctly resolved DNS request by a previously identified bot was also generated by the DGA or is legitimate traffic to find the C&C server.

The *DGA Discovery module* is given all NXDomain traffic collected within a specific time period at a DNS server. These domains are then clustered in two different ways. The first way starts by computing different statistical features of the observed domains. These features are n-gram features (the frequency distribution of n-grams for different n's), entropy-based features and structural domain features (like length or number of unique TLDs). Altogether, 33 different values are computed. With the help of the X-Means clustering algorithm [15], the different domains are grouped.

Based on the assumption that the same malware creates at least partially the same domain names, the second way is the creation of a bipartite graph consisting of host vertices on the one side and NXDomain vertices on the other side. If a host queried a certain NXDomain, these two vertices are connected with an edge. After some more improvements of the graph, the X-Means clustering is applied here as well, grouping NXDomains that have been requested by a similar set of hosts.

doi: 10.2313/NET-2014-08-1_04

To finalize the clustering phase, the two different sets of clusters are now merged by computing all possible intersections between both sets of clusters. All resulting clusters that are too small (the authors chose a minimum cluster size of 40) are dropped. All other clusters are kept for the next steps. As the authors lay focus on detecting new DGAs, they want to filter out all domain clusters generated by already known DGAs. This means that with the help of the DGA Classifier, it is determined if a certain cluster belongs to a known DGA with a certain probability and if that is the case, these clusters are dropped as well (but the IP addresses are added to a detection report).

The *DGA Classification and C&C Detection module* has two main purposes: It is supposed to determine if a specific set of NXDomains likely belongs to an already known DGA and it is supposed to tell if it is likely that a single active domain query performed by a host that already sent suspicious NXDomain requests also belongs to the botnet communication. The module takes a list of popular legitimate domains, a list of NXDomains generated by bots in a controlled environment and the newly discovered NXDomains from above as input.

The identification and comparison of NXDomain clusters is done with the help of the Alternating Decision Trees learning algorithm. The module compares each input with all known DGAs and returns for the most likely DGA a label with its name together with the probability that this assignment is correct.

The C&C detection works with Hidden Markov Models. One model is used per DGA. Before it can be used, it needs a training with a set of NXDomains generated by the same DGA. In active use, it is then given one successfully resolved domain at a time which was requested by a previously classified bot. The module calculates the probability that this particular domain name was also generated by the classified DGA. If it is above a certain probability threshold, it is branded a candidate C&C domain. The output of this detection module can then be used to maintain an IP blacklist to block bot communication with the C&C server.

Pleiades was tested within a time period of over two years observing 187,600 distinct hosts querying at least one NXDomain and 360,700 distinct NXDomains in total. Among other findings, six DGAs have been observed that do not belong to any known malware. An analysis of the DGA Classifier revealed that its detection rate lies at 99.7% with a false positive rate of 0.1%. The C&C server detection is not that successful: When choosing a false positive rate of 3%, the detection rate is above 91% for five out of six examined botnets. The detection rate for the last one (Boonana) is 27.67%. However, these results are still considerably better than the ones of previously presented systems above.

Altogether, Pleiades is a tool that is technically very advanced. The idea to search for DGA botnets by scanning the NXDomain traffic is implemented very well and the classification works with a very high detection rate. However, as the Boonana botnet shows, the detection of C&C servers – even though it works very well for most botnets – can be tricked: According to Antonakakis et al., Boonana creates pseudo-randomly third-level domains while the used second-level domains are owned by dynamic DNS providers [1]. This usage of two different domain levels obviously confused the Hidden Markov Model. The authors state that in a real-world scenario the detection rate of Boonana can be improved, though. Another consideration the authors make is that a DGA could deliberately produce a number of domain names that will never successfully resolve to a C&C IP. This is especially a problem if the same generator, but with a different seed, is used. That way, the learning process could be harder for the Hidden Markov Model.

There is no particular statement about Pleiades' performance. However, as it was used in a real environment for years, we can assume it is fast enough to handle the real-time DNS traffic. Additionally, no notes about privacy are made. But as no benign traffic is saved on purpose, privacy issues are very small for Pleiades. Nevertheless, anonymization of the data could add more privacy.

4.3 Predentifier

The Predentifier framework by Frosch et al. uses several passively collected DNS features and training sets to identify botnet communication [5]. In order to distinguish between benign traffic and malicious botnet traffic, 14 different features of DNS queries and their content in general are used. By applying the k-Nearest-Neighbor (kNN) technique, the traffic is classified.

Before the system can be used, it requires a training set of labeled benign and malicious domains which serves as reference to distinguish incoming traffic. For each domain, 14 different features are determined and saved. These features are lexical features of the domain (like the number of digits compared to length of domain), DNS answer-based features (like the number of returned IP addresses), IP address-based features (like the number of different ASNs of the IP addresses), zone-based features (like the TTL value) and WHOIS-based features (like the age of a domain). Frosch et al. state that 9 of these features have not been used ever before to find botnet activity in DNS traffic.

Now, the system is ready to classify new domains. For the domain of each DNS request, the feature set is determined and the kNN algorithm is applied to classify the domain. This works by considering the feature values as a vector, finding the k nearest neighbours is then done by calculating the Euclidean distance between two vectors. The new domain is then added to the group most of the k nearest neighbours belong to. In order to always be updated on the current status of benign and malicious domains, the authors propose to retrain the system every once in a while, so that the detection rate stays as high as possible. Predentifier does not employ any kind of clustering afterwards, it can only distinguish between benign and malicious.

A large part of the paper consists of the evaluation of the features as well as the whole framework. This evaluation shows that the application of all the new features significantly increases the detection rate (from 58.86% using only the five already "known" features to 94.19%). So, with the best choice of k being two, the overall detection rate lies at roughly 94.2% with a false positive rate of 8.7%.

doi: 10.2313/NET-2014-08-1_04

The training required by Predentifier must be repeated at times to keep the detection quality as high as possible which can be resource-intensive and lower the practicability of the approach. However, the authors make no statement about the performance of their system. Moreover, the initial input is crucial for the whole detection accuracy. Even though the system is flexible and can learn new types of malicious domains with the help of many different features, if the initial training set is badly chosen and lacks important domain groups, the detection rate can suffer.

The authors state that "only DNS answers are stored – information that is publicly available within the domain name system" and therefore, privacy is preserved. However, privacy is still an issue because the training sets still store DNS answers – even completely legitimate ones – with the IPs still visible. Storing the pages queried by an IP address can be secretly used for a wide variety of purposes – from personalized advertisement to identifying users accessing illegal pages. This means that the privacy is definitely affected by the framework. Such issues could be avoided, e.g., by employing an anonymization of the IPs with the help of hashing.

4.4 Privacy-Preserving Detection with Bloom Filters

Guerid et al. designed a framework using the NXDomains returned to bots, just like Pleiades [7]. The big difference of their system compared to all other presented is that they put emphasis on privacy issues. Before any processing for the botnet detection takes place, the input data is anonymized. The identification and clustering of bots and their C&C server is achieved by using Bloom Filters.

The first step of the framework is the anonymization of the IP addresses of the scanned traffic. This is done by hashing the addresses together with a periodically changing salt. This hash is subsequently called the "identifier" of the response. Then, the NXDomains are cleared of any responses that are easily identifiable as uninteresting, like queries with invalid characters or Top Level Domain. Afterwards, the remaining queries are sent to the *Community Construction Layer*.

The Bots Detection Module, first part of the *Community Construction Layer*, creates a Bloom Filter for each host that appears in the received NXDomains. A Bloom Filter is a data structure consisting of an array of bits. An input is hashed by a number of different hash functions and the bit in position in the array that corresponds to the hash result is set to 1. By default, it is 0. That way, all NXDomains queried by a certain host are stored in its filter (this framework here uses only one hash function per filter). If the filling ratio of the Bloom Filter is below 0.5% or above 10% (with the total size of the Bloom Filter being 1000 Bits), its host is dropped because it is considered uninteresting noise and it is no longer considered for any further processing.

The Bots Grouping Module, second part of the *Community Construction Layer*, is called by the Bots Detection Module and receives all newly collected identifiers and their Bloom Filters. It calculates how similar each host's queries are to each other to group bots by comparing their Bloom Filters. After grouping, all groups that contain more hosts than a specific paramater are passed on to the next module. This also means that any noise is filtered out, for example a host only with a misspelled, benign domain query.

In a last step, the *Malicious Domain Name Detection Layer* is responsible for the detection of the C&C servers. First, the C&C detection module is called and gets all the groups created by the Bots Grouping Module. It then creates new Bloom Filters for each host, this time with all the successfully resolved domain names. The filters within each group are then compared again to find overlapping domains. These domains are subsequently considered candidate C&C domains. Finally, in the Domain Name Validation Module, the system finds out if an identified domain could be benign and has no relations with botnet communication by computing a traffic dispersion indicator. If it is low, it means that a certain domain does not appear in many different bot groups which means that it is very likely a DGA domain (as different DGAs do not produce the same domain names).

Unfortunately, the evaluation by the authors lacks important information. No tests were performed to determine a reliable detection or false positive rate. Moreover, even though the paper was published in 2013, the DNS captures used were from 2009 and 2010. We assume that within that period, the landscape of botnets could have significantly changed so that the expressiveness of the evaluation is also questionable. An issue of the whole framework is that it requires the different bots of the same botnet to query the same domain names at roughly the same time. With the current implementation, the different bots could request the same domain with a larger time span in-between, for example more than one hour – as the authors state that they need a DNS history "of less than one hour [...] to detect malicious domain names" [7].

Nevertheless, performance-wise, the tool is able to work in real time as it is capable of processing half a million DNS queries per second (according to the authors, this is more than a typical DNS server has to handle). Moreover, the approach to preserve the users' privacy is commendable as there is no necessity to save the IP addresses of the DNS replies if only the C&C servers are supposed to be discovered. But if we want to identify the bots in the botnets, we are currently unable to do that with this implementation (although it would certainly be possible to monitor the IPs trying to access a previously identified C&C domain).

4.5 Anchor Domains

Gao et al. present a system that uses the correlation with predefined anchor domains to find new malicious domains [6]. Their system is not specifically designed to find botnets, but malicious domains in general. However, as their approach is different to the other ones and as they actually found new domains that are used presumably for botnet communication, it is mentioned here. The idea behind their framework is that it is very often the case that once a machine contacts a known malicious domain, it is very likely that it wants to contact other malicious domains at the same time or within a short period of time. These additional domains are supposed to be found.

doi: 10.2313/NET-2014-08-1_04

As input, the system requires beside the DNS traffic a set of known malicious domains A, the *anchor domains*. Additionally, a time window T_w must be specified. All domains that are requested half that time before or half that time after a malicious domain is queried are analyzed. All the domains d collected around one anchor domain are called a segment s. The set S contains all segments. In the next step, by calculating two values m_{tf} and m_{idf}, the TF-IDF (term frequency – inverse document frequency) metric is applied. It measures how many times a domain name occurs while taking into account that there are popular domains that are uninteresting for us. The term frequency m_{tf} of a domain d and a segment s holds the number of appearances of the domain in that segment. The inverse document frequency $m_{idf} = |S|/|\{s \in S : d \in s\}|$ mitigates the impact of legitimate, well-known domains that are likely queried often by all hosts. If both values are above certain thresholds T_{tf} and T_{idf}, the corresponding domain is kept. All other domains are disregarded.

In the next step, the clustering is done with the help of the X-Means clustering [15] algorithm and the *pattern of co-occurrence* with the anchor domain (this means that domains appearing in the same segments may be more likely grouped than domains not having that in common). After splitting the segments in more subsegments and applying two more filters for further improvement of the results, the clustering is finished and a set of clusters is presented that should be analyzed in more detail.

In an evaluation of their approach, Gao et al. detected 6890 previously unknown malicious domains, among them also domains supposedly generated by a DGA. Each anchor domain could be expanded to an average of 53 newly detected malicious domains. Their false positive rate lies at 3.6%. As each anchor domain can be treated independently, their approach is easily parallelizable, leading to a good performance.

The largest issue of that framework is that it is only as good as the list of anchor domains it is provided with. Any malicious domain that stands in no correlation with any of the anchor domains will never be detected. This is especially a problem with DGAs, as one of their core feature is to regularly generate new domains and to never use the old ones again. So, either the blacklist must be updated very quickly or such DGA-based malware will never be found. That is why this approach is not perfectly suited to find botnets with these DGA-based DNS queries. No specific measures are taken to protect the privacy and anonymize the collected input, as well.

5. CONCLUSION

In this paper, we gave an overview on recently published systems to detect botnets and their C&C servers by monitoring the DNS traffic. This survey shows that there are many different approaches. Most of them show a high detection rate for botnets using DGAs or fast-flux techniques of up to 99.7%. Additionally, the performance of most frameworks was also proven in a live scenario at large DNS nodes which makes them actually deployable. Nevertheless, except for one system, privacy issues are largely disregarded. The frameworks handle a large amount of data, therefore, they must take care of the protection of a user's privacy.

However, the development of botnet malware will never stop and malware developers will also have noticed the effort taken to detect botnets – be it by DNS or otherwise. This means that the development of new malware trying to evade and trick the different mechanisms presented here will continue. Therefore, the progress of botnet development must be closely watched to be able to quickly react on new communication techniques and to develop new frameworks to stop them.

6. REFERENCES

[1] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon. From throw-away traffic to bots: Detecting the rise of dga-based malware. In *Proceedings of the 21st USENIX security symposium*, 2012.

[2] A. Caglayan, M. Toothaker, D. Drapeau, D. Burke, and G. Eaton. Real-time detection of fast flux service networks. In *Conference For Homeland Security, 2009. CATCH'09. Cybersecurity Applications & Technology*, pages 285–292. IEEE, 2009.

[3] V. Cardellini, M. Colajanni, and S. Y. Philip. Dynamic load balancing on web-server systems. *IEEE Internet computing*, 3(3):28–39, 1999.

[4] D. Dittrich and S. Dietrich. P2p as botnet command and control: a deeper insight. In *Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on*, pages 41–48. IEEE, 2008.

[5] T. Frosch, M. Kührer, and T. Holz. Predentifier: Detecting botnet c&c domains from passive dns data.

[6] H. Gao, V. Yegneswaran, Y. Chen, P. Porras, S. Ghosh, J. Jiang, and H. Duan. An empirical reexamination of global dns behavior. In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, pages 267–278. ACM, 2013.

[7] H. Guerid, K. Mittig, and A. Serhrouchni. Privacy-preserving domain-flux botnet detection in a large scale network. In *Communication Systems and Networks (COMSNETS), 2013 Fifth International Conference on*, pages 1–9. IEEE, 2013.

[8] T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling. Measuring and detecting fast-flux service networks. In *NDSS*, 2008.

[9] E. J. Kartaltepe, J. A. Morales, S. Xu, and R. Sandhu. Social network-based botnet command-and-control: emerging threats and countermeasures. In *Applied Cryptography and Network Security*, pages 511–528. Springer, 2010.

[10] B. Krishnamurthy, C. Wills, and Y. Zhang. On the use and performance of content distribution networks. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pages 169–182. ACM, 2001.

[11] F. Leder and T. Werner. Know your enemy: Containing conficker. *The Honeynet Project, University of Bonn, Germany, Tech. Rep*, 2009.

[12] P. Lin. Anatomy of the mega-d takedown. *Network Security*, 2009(12):4–7, 2009.

[13] P. Mockapetris. RFC 1034 - Domain Names - Concepts and Facilities, 1987.

[14] P. Mockapetris. RFC 1035 - Domain Names -

Implementation and Specification, 1987.

[15] D. Pelleg, A. W. Moore, et al. X-means: Extending k-means with efficient estimation of the number of clusters. In *ICML*, pages 727–734, 2000.

[16] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna. Your botnet is my botnet: analysis of a botnet takeover. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 635–647. ACM, 2009.

doi: 10.2313/NET-2014-08-1_04

Using GPUs for Packet-processing

Beschleunigte Paketverarbeitung mit Hilfe von GPUs

Thomas Schultz
Betreuer: Daniel Raumer
Seminar Future Internet SS2014
Lehrstuhl für Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: schultz@in.tum.de

KURZFASSUNG

Software-Router sind den Hardware-Routern in Bereichen wie Flexibilität und Erweiterbarkeit weit voraus, doch haben sie trotz moderner Hardware immer noch Probleme das Leistungspotential auszuschöpfen. Um effizientere Systeme zu entwickeln, versucht man die Leistung von Grafikprozessoren hierfür zu nutzen. Im folgender Abhandlung wird die Motivation für diesen Schritt als auch die Hindernisse auf dem Weg dorthin beschreiben. Anschließend werden zwei Implementierung auf Basis dieser Technik vorgestellt.

Schlüsselworte

software router, gpgpu

1. MOTIVATION:

Computernetzwerke erfüllen heutzutage eine wichtige Rolle in der globalen Kommunikation. Um die stetig wachsenden Datenmengen zu bewältigen werden immer neue Techniken benötigt. Auf der eine Seite betrifft das die reine Übertragungstechnik, basierend auf kabellosem Funk oder kabelgebunde Kupfer- bzw. Glasfaserkabel. Über Glasfaserverbindungen lassen sich dabei die höchsten Datenraten erzielen. Aktuell wird an Linkgeschwindigkeiten von 100 Gbit/s gearbeitet [1]. Auf der anderen Seite muss auch die Verarbeitung der Daten zwischen den Kommunikationspartnern schneller werden, um diese Geschwindigkeiten auch auszunutzen. Diese Verarbeitung, das Packet-processing, ist von fundamentaler Bedeutung für die Infrastruktur in großen Netzwerken wie zum Beispiel dem Internet. So benutzt jeder die großen Backbone-Netzwerke der Internetprovider, wenn er über seinen Internetanschluss Daten empfängt oder überträgt.

Gerade wenn es um kommerzielle Ziele geht, ist es von Vorteil, wenn sich das Verhalten dieses Netzes besser kontrollieren lässt. Dadurch könnten vertraglich festgelegte Leistungen gegen einen Aufpreis zugesichert werden. Gerade in jüngerer Vergangenheit, bei den Diskussionen um die Netzneutralität, offenbarte die Deutsche Telekom Pläne, wonach sie gezielt eigene Dienste wie IPTV (T-Entertain [9]) bevorzugen wollte. Derzeitige Systeme ermöglichen aber nur begrenzten Einfluss auf die Verarbeitung der Datenpakete. Die Betreiber sind hier an den gebotenen Funktionsumfang der Hersteller gebunden. Ohne diese lassen sich Änderungen an vorhandenen Systemen sehr schwer oder gar nicht durchsetzen. An dieser Stelle sind Software-Router proprietären Hardware-Routern, zum Beispiel von Cisco, in der Konfigurierbarkeit überlegen. Ihre Software ist in weiten Teilen unabhängig von der Hardware und kann so leicht erweitert oder gar ausgetauscht werden, um neue Funktionen zu implementieren.

Software-Router sind im Prinzip normale Computer, wie man sie im Büro oder zu Hause stehen hat. Ihr Vorteil besteht zu großen Teil in ihrer Flexibilität. Die benötigte Hardware ist standardisiert, überall zu erwerben und preislich um einiges günstiger als spezielle Netzwerk-Hardware. Zum Anderen ist die Paketverarbeitung in Software implementiert, die in den meisten Fällen quell-offen, also *open-source*, verfügbar ist. Damit lassen sich die Kosten einsparen und eigene Anpassungen bzw. Optimierungen für das Anwendungsgebiet vornehmen.

Das Problem dabei ist, dass Software-Router trotz der heutigen Rechensystem noch immer nicht die Leistung erbringen, um kommerziell in den Markt der etablierten Hardware-Router einzudringen. Immer neuere Forschungsprojekte belegen das große Interesse und Potential von Software-Routern [1] [2] [4]. Darunter sind ein paar Arbeiten, die sich mit parallelisierter Verarbeitung von Paketen mit Hilfe von Grafikkarten befassen. Dass moderne Grafikkartenarchitekturen nicht nur für Computerspiele geeignet sind, beweist die Top500 Liste [2] der weltweiten Supercomputer. In der Top10 finden sich bereits mehrere Systeme, die zur Steigerung der Rechenleistung auf Beschleunigungskarten auf Basis von Grafikprozessoren (GPUs) setzen. Zusätzlich entwickelt sich die Leistungsfähigkeit von GPUs im Gegensatz zu herkömmlichen Prozessoren (CPUs) schneller [7].
Im folgenden soll geklärt werden, warum sich GPUs für Paketverarbeitung eignen und wie man sie richtig einsetzt.

2. SYSTEM-ARCHITEKTUR

Um zu verstehen wie GPUs für die Verarbeitung von Netzwerkpaketen verwendet werden können, muss das gesamte System betrachtet werden. Ein einfacher Software-Router besteht, wie jeder PC, aus einem oder mehrerer Prozessoren, einem Arbeitsspeicher und geeigneten Netzwerkkarten. Dabei kann man die zugrundeliegende Hardware in zwei Klassen unterteilen: homogene und heterogene Systeme.

[1] IEEE P802.3bm Task Force `http://www.ieee802.org/3/bm/`

[2] Top500 Supercomputers: `http://www.top500.org/`

doi: 10.2313/NET-2014-08-1_05

2.1 Homogene Systeme

In einem homogene System befinden sich nur Komponenten einer einzelnen bestimmten Architektur. Die meist verbreitete und somit bekannteste ist die x86-Architektur von Intel. Diese basiert auf Intels Mikroprozessor 8086 aus dem Jahr 1978 und erbte dessen Namensschema. Trotz vieler Erweiterungen ist sie heute immer noch Basis moderner Mikroprozessoren von Intel sowie dem Konkurrenten AMD. Nebenbei gibt es auch andere bekannte Architekturen, zum Beispiel ARM, welche heutzutage in fast jedem Smartphone verwendet wird.
Ein Software-Router auf einem solchen System ist folglich an die verfügbaren Fähigkeiten dieser Architektur gebunden. Die x86-Architektur ist historisch bedingt auf komplexe, seriell zu verarbeitende Befehle optimiert. Multicore-Unterstützung wurde erst Anfang der 2000 Jahre relevant. Um den Sachverhalt zu vereinfachen, könnte man die x86-Architektur als gutes Allzweckwerkzeug bezeichnen, welches für die meisten Aufgaben gut, jedoch für wenig optimal geeignet ist.

2.2 Heterogene Systeme

Mit aufkommenden Bedarf an Leistungsverbesserungen wurde der Einsatz von spezielleren Architekturen vorangetrieben. In einem heterogenen System sind mehrere Komponenten unterschiedlicher Bauarten verbunden, um je nach Anforderung auf diese spezielle Hardware zurück zu greifen. Das beste Beispiel hierfür sind moderne Heimcomputer oder gar Notebooks. Neben einem Prozessor sind meistens dedizierte, eigenständige Grafikkarten verbaut. Diese vornehmlich für Spielzwecke entwickelten Ergänzungskarten haben sich mittlerweile auch einen Platz im wissenschaftlichen Umfeld geschaffen. Bei Verwendung von GPUs für Rechenoperationen, anstatt zur Grafikausgabe spricht man auch vom Begriff der *General Purpose Computation on Graphics Processing Unit* kurz GPGPU.
Durch ihre rapide Entwicklung und immensen theoretischen Rechenleistung, die um mehrere Faktoren größer ausfällt als die von leistungsstarken CPUs der selben Generation sind sie ein beliebtes Werkzeug. Seitdem NVIDIA und AMD (ehemals ATI) diesen Trend erkannt haben, bieten beide dementsprechend leicht modifizierte Karten für diesen Markt an. Nebenbei wurden auch angepasste Programmierschnittstelle (dazu später mehr in 2.4.1) veröffentlicht.

2.3 Software-Router

Betrachtet man einen Software-Router zunächst als Black-Box, benötigt dieser drei Elemente: die Netzwerkports, einen Verarbeitungspuffer sowie eine Verarbeitungslogik. Bildet man das auf einen Computer ab erhalten wir die wichtigen Bestandteile eine Software-Routers: Netzwerkinterfaces (NICs), Arbeitsspeicher (RAM) und Prozessoren. Wie der Name verrät, erledigt ein Software-Router seine Funktionalität durch den Kernel des Betriebssystems oder Programmen im Userspace. Dieser Schritt erfordert jedoch, dass jedes Netzwerkpaket im Arbeitsspeicher liegt, damit die CPU auf die Daten bzw. Header zugreifen kann. Folglich entstehen bei der Verarbeitung drei potentielle Schwachstellen:

1. I/O: Die Netzwerkkarte muss immer alle Pakete annehmen können
2. RAM: Lese- und Schreiboperationen müssen schnell genug geschehen

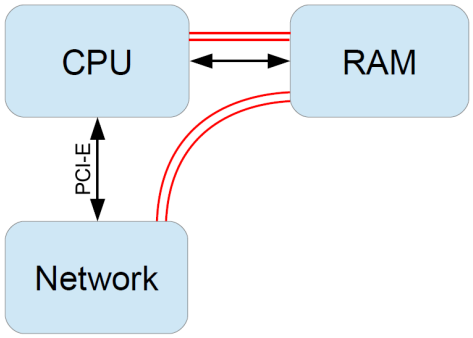


Abbildung 1: Vereinfachte Darstellung der Speicherzugriffe

3. CPU: Zügige Bearbeitung zur Vermeidung von Paket-Staus

Um einen groben Überblick über die Problematik zu erhalten, wird auf alle drei Punkte kurz eingegangen.

2.3.1 I/O Bandbreite

Heutige x86-Systeme unterstützen in der Regel den PCI-Express [3] Standard. Diese Schnittstelle wird zur Anbindung zusätzlicher Hardware in einem Computer verwendet. Grafikkarten, Soundkarten, spezielle SSDs und auch Netzwerkkarten verwenden diese Technik zur Kommunikation. Seit der Einführung im Jahr 2003 hat sich die mögliche Bandbreite dieser Schnittstelle stetig gesteigert. Zusätzlich ist die Ansteuerungslogik mittlerweile vom Chipsatz in die CPU selber gewandert. Um größere Flexibilität zu erlauben, ist der Standard modular aufgebaut. Es gibt verschieden große Steckplätze auf dem Mainboard, die jeweils mit 1, 2, 4, 8 oder 16 sogenannten Lanes angeschlossen werden. Dabei ist zu beachten, dass physikalisch und elektronisch Abwärtskompatibilität besteht, d.h eine x1 Karte passt in einen x16, umgekehrt jedoch nicht. Der aktuelle Standard, PCI-Express 3.0, ist in der Lage auf einem vollen 16-Lanes Steckplatz knapp 16 GB/s voll-duplex zu übertragen.
Eine über PCI-Express angeschlossene Netzwerkkarte muss also mindestens die Bandbreite für ihre Linkgeschwindigkeit samt Overhead zur Verfügung haben, oder sie wird bereits in der Performance gebremst. Für eine einzelne Netzwerkkarte mag dies noch kein Limit darstellen, kann aber bei größerer Anzahl und je nach Geschwindigkeit bzw Anzahl physikalischer Anschlüse (Ports) zu einem Problem werden. Jedes System kann nur eine bestimmte Zahl an PCI-Express-Lanes zur Verfügung stellen, so dass auch hier die maximale Bandbreite auf alle Erweiterungskarten aufgeteilt werden muss. Je nach Anzahl kann es passieren, dass die Steckplätze auf dem Mainboard nur mit der Hälfte oder weniger Lanes betrieben werden.

2.3.2 Speicher

In einer Von-Neumann-Architektur müssen alle zu verarbeitenden Daten im Arbeitsspeicher liegen. Seitdem Intel mit der Core-i Generation mit AMD gleichgezogen hat, und den RAM-Controller in die CPU integriert hat, verwenden beide die gleiche Topologie. Die einzelnen Arbeitsspeicher Module

[3] PCI-Express Spezification: `http://www.pcisig.com/specifications/pciexpress/`

doi: 10.2313/NET-2014-08-1_05

sind direkt mit dem Controller verbunden und müssen nicht mehr wie früher über einen gemeinsamen Systembus angesprochen werden.
Durch diverse Techniken, darunter DDR3 (DDR4 ist bereits für 2014 angekündigt) und die Verwendung von zum Beispiel Quad-Channel (paralleler Zugriff auf vier Speichermodule) ist die mögliche Bandbreite stetig gewachsen. Dennoch ist der Arbeitsspeicher eine kritische Systemkomponente eines Software-Router. Selbst im optimalen Fall, siehe Abbildung 1 führt ein Paket zu vier Zugriffen auf dem Speicher-Controller: Netzwerkkarte schreiben, CPU lesen, CPU schreiben, Netzwerkkarte lesen. Zwar lässt sich die Datenmenge dieser Zugriffe optimieren, falls zum Beispiel nicht das komplette Paket sondern nur der Header benötigt wird, jedoch gibt es keinen generellen Ausweg aus dieser Problematik.

2.3.3 Prozessor

In einen Software-Router müssen alle Entscheidungen und Berechnungen von einem Prozessor erledigt werden. Dieser muss entweder Teile oder auch das gesamte Paket lesen, um die konfigurierten Regeln anzuwenden. Für einen einfachen IP-Router besteht diese Aufgabe mindestens aus einem Nachschlag in der Routing-Tabelle und dem Herabsetzten des TTL-Wertes.
Der große Vorteil gegenüber Hardware-Routern ist jedoch die große Flexibilität der Software. Dadurch sind auch kompliziertere Anwendungsfälle einfacher zu realisieren. Virtual-Private-Networks (VPNs) zum Beispiel benötigen mehr Aufwand durch die Verkapselung von Datenpaketen und den nötigen kryptografischen Rechenoperationen. Je nach Komplexität der Konfiguration kann der Rechenaufwand für die CPU um Faktoren steigen, sodass diese die Performance des Software-Routers beeinträchtigt.
Eine Prozessor mit nur einem Kern muss alle empfangenen Pakete bearbeiten. Die Zeit die er dafür maximal benötigen darf, ist durch die Linkgeschwindigkeit und die Paketgröße gegeben. Verteilt man die Arbeit auf mehrere Kerne erhöht sich diese Zeit, jedoch kommen sich die einzelnen Kerne bei Zugriff auf Speicher bzw. dem gemeinsamen Cache in die Quere.

2.4 GPGPU

Will man nun einen Software-Router mit Hilfe von GPUs beschleunigen, sind einige wichtige Aspekte zu beachten. In erster Linie sind das die fundamentalen Unterschiede zwischen einem CPU- und einem GPU-Kern. Im Folgenden werden beide Begriffe auch für den physikalischen Chip, dem sogenannten *Die* verwendet.

Bei einer modernen CPU befinden sich bereits mehrere Kerne auf einem Chip (z.B. Intel XEON Server CPUs mit bis zu 15[4]). Neben diesen Rechenkernen wird ein Großteil der Transistoren für den Cache benötigt. Dieser Cache wird zur Pufferung von Daten aus dem Arbeitsspeicher als auch für die Synchronisation unter den CPU-Kernen verwendet. Moderne CPUs verwenden eine Vielzahl von Funktionen (Pipelining, Out-of-Order Execution, etc) um die Ausführung von Befehlen zu beschleunigen. Für diese Optimierungen werden zusätzliche Transistoren neben den eigentlichen Recheneinheiten, den *arithmetic logic units* (ALUs), benötigt, wodurch eine CPU noch komplexer wird.

Eine GPU ist dagegen anders aufgebaut. Eine handelsübliche Grafikkarte verfügt auf ihrer Platine über einen eigenen dedizierten Speicher, der im Vergleich zu dem Arbeitsspeicher des Hosts eine größere Bandbreite besitzt. Der Großteil der GPU selber besteht aus vielen simplen Rechenkernen die für paralleles Arbeiten ausgelegt sind. Diese SIMD-Einheiten (Single Instruction, Multiple Data) führen eine Operation auf einem Vektor an Daten aus. Aus diesem Grund verfügen GPU auch über eine deutlich höhere theoretische Rechenkraft als CPUs. Des Weiteren bedient man sich bei Grafikkarten eines weiteren Tricks. Möchte die CPU auf Daten zugreifen, die zur Zeit nicht mehr im Cache liegen, verliert man wertvolle Zyklen bis die Daten aus dem Arbeitsspeicher geladen wurden. Alternativ kommt ein anderer, wartender Thread an die Reihe. So ein Kontextwechsel ist allerdings mit Overhead verbunden, da alle Register zurück in den Cache geschrieben und neue Daten geladen werden müssen.
Bei GPUs hingegen führt ein Wechsel zu kaum Verlusten, sodass bei Speicherzugriff einfach ein anderer Thread ausgeführt wird. Diese Technik nennt sich Latency-Hiding. Damit eine GPU effektiv ausgelastet ist, müssen also immer mehr Threads vorhanden sein als verarbeitet werden. Bei Kernzahlen um die 2500 (NVIDIA GK110 bis 2880 [6]), können das unter Umständen zehntausende sein.

2.4.1 Verwendung von GPUs

Obwohl der Einsatz von Grafikkarten zur Beschleunigung mittlerweile weit verbreitet ist, ist ihre Verwendung umständlich. Programme laufen grundsätzlich erst auf dem Prozessor des Hosts. Um die Rechenleistung der GPU zu nutzen, müssen die Daten auf den getrennten Speicher der Beschleunigungskarte kopiert, deren Ausführung initiiert werden und nach Abschluss wieder in der Arbeitsspeicher des Hosts zurückgeschrieben werden. Dabei wird sowohl der Speicher-Controller sowie die PCI-Express Schnittstelle belastet. Bei diesen Kopiervorgängen muss darauf geachtet werden, dass entsprechende Daten im Cache der CPU als ungültig erklärt werden, damit die Datenkonsistenz gewährleistet bleibt.

Für Programmierer von GPU-beschleunigtem Code stehen diverse Frameworks zur Verfügung. Diese Frameworks stellen dem Anwender eine abstrahierte Schnittstelle zwischen der Hardware und gängigen Programmiersprachen zur Verfügung. Als bekannteste Vertreter sind hier NVIDIA's CUDA und OpenCL zu nennen. Beide unterscheiden sich hinsichtlich ihrer Implementierung und der unterstützen Hardware, sind aber vom Prinzip sehr ähnlich. Während CUDA nur auf Produkten von NVIDIA selber läuft, ist OpenCL unabhängig von der verwendeten Plattform. Im Folgendem wird das Prinzip dieser Schnittstellen anhand von OpenCL erklärt.

Mit OpenCL[13] lassen sich mehrere Ressourcen in einem System vereinen. Dazu wird die Hardware nach OpenCL fähigen Geräten durchsucht. Diese sind in der Regel in unabhängige Recheneinheiten aufgeteilt. Das Hauptprogramm hat nun die Aufgabe, die Verwendung dieser Recheneinheiten zu steuern. Dazu werden die Daten in sogenannte Work-Items aufgeteilt und an die Recheneinheiten verteilt. Für

[4] Intel Xeon E7 Family: `http://ark.intel.com/products/family/78584`

doi: 10.2313/NET-2014-08-1_05

die Ausführung werden sogenannte Kernels aus dem Programmcode erzeugt, die dann die Work-Items verarbeiten. Dieses Hauptprogramm kann in beliebiger Sprache geschrieben sein, die OpenCL unterstützt. Der OpenCL Code, angelehnt an C99, wird erst während der Laufzeit für die verwendete Hardware compiliert und dann auf dieser ausgeführt.

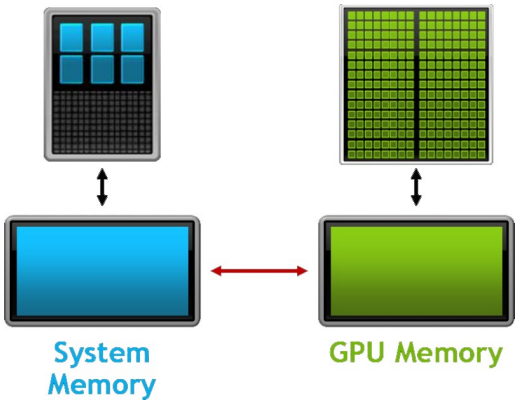


Abbildung 2: Vorherrschendes Speicherkonzept (Quelle [10])

2.4.2 *Unified Memory*

Um die Nachteile dieser getrennten Speicherbereiche zu beseitigen, strebt man Systeme auf Basis gemeinsamer Speicher (Unified-Memory) an. Bei einem solchen System können CPU und GPU auf einen gemeinsamen Speicher zugreifen. Dadurch werden Kopiervorgänge zwischen beiden Einheiten unnötig. Auch aus Sicht der Programmierers gestaltet sich die Verwendung einfacher, da statt den Daten selber einfach Pointer übergeben werden können. In Abbildung 2 ist das zur Zeit verwendete Konzept dargestellt. Alle Daten müssen hier zwischen den beiden Speicherblöcken übertragen werden, hier angedeutet durch den roten Pfeil. Abbildung 3 hingegen besitzt nur einen Speicherblock auf den von beiden zugegriffen werden kann.

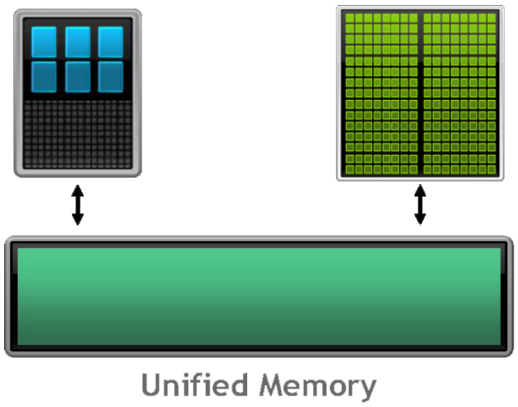


Abbildung 3: Unified Memory Konzept (Quelle [10])

In der Praxis sind solche Systeme selten anzutreffen, da die Architektur schwierig zu implementieren ist. Um Kohärenz zu bewahren müssen sich CPU und GPU bei Zugriff auf den gemeinsamen Speicher absprechen. Der gemeinsame Speicher-Controller muss mit seiner Bandbreite beide Kerne gleichzeitig bedienen. Verwendet man, wie üblich, gewöhnliche Speichertechnologie wie DDR3 entsteht hier ein Engpass. Dadurch sind die eigenständigen Beschleunigungskarten in Bereich der Speicherbandbreite deutlich im Vorteil. Hier müssen die Daten zwar erst per PCI-Express auf diesen gelangen, doch von dort erfolgt der Zugriff deutlich schneller. Das wiederum heißt Softwareentwickler müssen sich hier weniger Sorgen um Speicherzugriffe während der Bearbeitung machen.

Auf dem Markt gibt es dennoch Vertreter dieser Art. Seit dem AMD den Grafikkartenhersteller ATI aufgekauft hat, produzieren sie Mikroprozessoren auf denen neben x86 basierten CPU-Kernen auch GPU-Kerne verbaut sind. Dieses Prozessorkonzept verkauft AMD unter den Namen *Heterogeneous System Architecture* kurz HSA [5]. Die von AMD *Accelerated Processing Units* (APUs) genannten Modelle verwenden den sonst nur für die CPU ausgelegten Arbeitsspeicher für beide Arten von Kernen.

Die Leistung dieser Kombination kommt allerdings noch nicht an die Leistung von eigenständigen Beschleunigungskarten heran. Dieser Umstand ist auch nicht sonderlich verwunderlich, wenn man die thermische Verlustleistung betrachtet. Während moderne Grafikkarten auf komplizierte Kühlkonzepte setzen, um den Chip, Speicher und deren Stromversorgung innerhalb der thermischen Grenzen z halten, muss sich eine integrierte Grafikeinheit diese Ressourcen mit dem Hauptprozessor teilen. Trotzdem ist die Idee der Vereinigung beider Architekturen auf Chip-Level statt auf System-Level ein vielversprechender Ansatz. Dadurch können PCI-Express Steckplätze statt für Beschleunigungskarten für Netzwerkkarten verwendet werden, während die Rechenleistung durch Multi-Sockel Systeme bereitgestellt wird. Da es sich allerdings um eine relativ jungen Entwicklung handelt fehlen bisherige Ergebnisse in Bezug auf Software-Router.

3. PACKET-PROCESSING

Die Verarbeitung von Netzwerkpaketen ist ein relativ junges Einsatzgebiet für Beschleunigungskarten auf Basis von GPUs. Ein Leistungssteigerung kann nur erreicht werden, wenn sich die Berechnungen genügend parallelisieren lassen. Im Fall von Software-Routern, lässt sich die Verarbeitung von einzelnen Paketen sehr gut parallelisieren. In den folgenden Abschnitten wird nun auf die Besonderheiten dieses Einsatzgebiets eingegangen.

3.1 Zero-Copy Mechanismen

Eine Möglichkeit die interne Verarbeitung zu beschleunigen sind sogenannte Zero-Copy Mechanismen. Im Prinzip versucht man mit dieser Strategie, unnötiges Kopieren der Daten von einer Stelle im Speicher zu einer anderen zu vermeiden. Wie bereits im Abschnitt 2.3.2 beschrieben, führen solche Speicherzugriffe zu Bandbreitenverbrauch sowie größere Verzögerung in der Verarbeitung.
Benutzt man nun separate Grafikkarten mit eigenständigem Speicher muss man sich zwangsläufig Gedanken machen, welche Daten man in den Kopiervorgang einbezieht. Es gibt bereits einige Tools [8], die diese Technik einsetzen und gezeigt haben, dass hier ein großes Potential liegt.

3.2 Paralleles Bearbeiten

Der Grund warum die meiste Software von Mehrkern-Systemen nur begrenzt profitiert, sind Daten- oder Kontrollabhängigkeiten. Sobald die Ausführung von Code-Segmenten

doi: 10.2313/NET-2014-08-1_05

in irgendeiner Art von dem Ergebnis vorheriger Berechnungen abhängen, muss diese Information erst intern kommuniziert werden. Je mehr Kerne verbaut sind, desto größer wird generell dieser Kommunikations-Overhead und bedingt so nur eine mäßige Skalierung.
Bei Netzwerkpaketen ist in der Regel ein Datenpaket unabhängig von seinem Vorgänger. Damit lässt sich der Rechenaufwand auf mehrere Rechenkerne aufteilen, ohne dass es zu Synchronisationsproblemen führt. Durch diesen Umstand lässt sich die Verarbeitung durch Grafikkarten beschleunigen.

Blockweise Bearbeitung

Wie bereits erwähnt, sind GPUs auf die parallele Ausführung optimiert. Im Gegensatz zu den meisten wissenschaftlichen Aufgaben die auf GPGPU laufen, enthalten Netzwerkpakete nur relativ wenig Daten, die nur kurze zur Bearbeitung in der GPU verweilen. Initiiert man also für jedes eingehende Paket einen GPU-Thread wird viel Zeit und Bandbreite für die Koordination verbraucht [3]. Überträgt man hingegen in einem Aufruf mehrere Datensätze, so kann man den Mehraufwand pro einzelnes Paket stark senken.
Als Konsequenz daraus, erhält man erst einen Vorteil, wenn man Paket-Blöcke erstellt. Auf diese Weise wird die Auslastung der Systemschnittstellen reduziert und gleichzeitig kann garantiert werden, dass die GPU effizient ausgenutzt wird. Damit ein Software-Router also von einer Grafikkarte profitieren kann, müssen mehrere Netzwerkpakete in einem Puffer gesammelt und erst anschließend als ein Block an die GPU übergeben werden.

Messungen aus [11] mit verschiedenen Blockgrößen ergeben einen Leistungszuwachs ab ein paar hundert Paketen pro Block. Dabei ist zu beachten, dass dieser experimentell erprobte Wert von dem Funktionsumfang abhängt. Ebenso ist ein klarer Trend festzustellen: je größer die Blöcke werden, desto größer fällt der Leistungszuwachs aus. Auch die Daten aus [3] bestätigen die Größenordnung dieses Wertes.

Verzögerung

Das Zusammenfassen von Netzwerkpaketen vor der Verarbeitung erzeugt jedoch Zeit-bedingt eine gewisse Verzögerung. Um einen guten Kompromiss zwischen zeitlicher Verzögerung und Verarbeitungsgeschwindigkeit zu erreichen, ist die Blockgröße dementsprechend auszulegen. Dabei begünstigen schnellere Übertragungsgeschwindigkeiten das Problem, da pro Pakete geringere Zeitintervalle benötigt werden.
Zusätzlich zu der Größe der einzelnen Blöcke sollte auch ein Time-out konfiguriert werden. Für den Fall, das keine Pakete mehr ankommen, würde der letzte Block solange wartend verweilen, bis die nötige Anzahl erreicht ist. Durch hinzufügen eine Time-outs kann sicher gestellt werden, dass die Verzögerung auch in diesem Fall nicht höher ausfällt.

Die Autoren von [11] haben in ihren Test ebenso die Verzögerung gemessen. Hierfür stellten sie Messung der Paketumlaufzeit (Round Trip Time or RTT) mit ausschließlich Verarbeitung auf der CPU sowie mit Blöcken variabler Größen auf der GPU an. Bei eine Blockgröße von 1024 ergaben sich Werte von knapp dem zehnfachen von denen auf der CPU. Reduzierung dieser Zeit ist nur auf Kosten des Durchsatz möglich.

Reihenfolge

Ein weiteres Problem paralleler Verarbeitung ist die Reihenfolge. In einer strikt seriellen Verarbeitung wird wie in einer FIFO-Warteschlange alle eingehenden Pakete der Reihe nach bearbeitet. Das wiederum heißt, die Reihenfolge der Pakete, beispielhaft einer TCP Verbindung, bleibt erhalten. Werden die Pakete hingegen auf verschiedene Threads auf der Grafikkarte aufgeteilt, so ist die Reihenfolge ihrer Bearbeitung unbestimmt. Soll neben einem IP-Router auch noch weitere Funktionalitäten wie einer Firewall oder ähnliches integriert werden, muss dafür gesorgt werden, dass die Verarbeitung das gleiche Ergebnis wie die serielle Verarbeitung liefert.
Ein Möglichkeit die Reihenfolge nicht zu verändern, ist die Zuteilung eingehender Pakete anhand ihrer Ursprungs- und Ziel-Adresse. Gelangen alle Pakete mit den gleichen Eigenschaften in einen Block, kann garantiert werden, dass die Reihenfolge erhalten bleibt. In der Regel wird dies durch eine geeignete Hash-Funktion gelöst, die alle möglichen Pakete auf die zu Verfügung stehenden Ressourcen aufteilt.

4. BESPIELE

Leistungssteuerung und Messung von Software-Routern ist eine komplexe Angelegenheit. Zum Einen werden ständig neue Technologie entwickelt die wiederum bisherige Ergebnisse relativieren. Zum Anderen ist die verwendete Software, darunter Betriebssystem, Treiber und Applikation selber, in ständigem Wandel. Aus diesem Grund kann es vorkommen, dass verschiedene Veröffentlichungen unterschiedliche Erkenntnissen liefern. Die Optionen sind meist so vielfältig, dass trotz genannter Details oft kein direkter Vergleich untereinander möglich ist.
Will man zusätzlich noch die Effektivität von GPUs bewerten, ist viel Erfahrung mit Software-Routern sowie der Hardware nötig. Aus diesem Grund gibt es relativ wenig Vielfalt bei diesen Arbeiten, wovon hier kurz zwei Ansätze vorgestellt werden.

4.1 Snap und Click

Snap [11] ist eine GPU-basierte Erweiterung für Click (*click modular router* [5]). Click ist ein Framework für Software-Router Konfigurationen für Linux. Click basiert auf einer graphenorientierte Beschreibung. Der Router besteht aus einer Anordnung von Knoten, den Elementen, welche die Funktionen repräsentieren und Verbindungen, welche mögliche Pfade für Netzwerkpakete darstellen. Abbildung 4 veranschaulicht einen simplen Graphen, bei dem alle am Interface *eth0* ankommenden Pakete gezählt und anschließend verworfen werden. Zur Erstellung einer Konfiguration wird eine Textdatei genutzt, die den Graphen beschreibt.

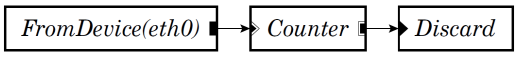


Abbildung 4: Router zählt Pakete und verwirft alle (Quelle: [4])

Click alleine besitzt keine Unterstützung für die Verwendung von GPUs, weshalb Snap entwickelt wurde. Snap fügt eine neu Art von Elementen hinzu, die direkt auf von der GPU arbeiten. Zu diesem Zweck wurde der Quellcode von Click

[5] Click Modular Router: `http://www.read.cs.ucla.edu/click/click`

doi: 10.2313/NET-2014-08-1_05

nur an wenigen Stellen modifiziert und hauptsächlich um die neuen Elemente ergänzt.
Um die Flexibilität von Click zu bewahren stehen weiterhin alle ursprünglichen Elemente zur Verfügung. Sollen Pakete jetzt auf der GPU verarbeitet werden gibt es zwei Schnittstellen: Host-to-GPU und GPU-to-Host. Dazwischen werden Elemente platziert die direkt auf der GPU ablaufen. Snap verwendet die erste in Abschnitt 3.2 beschriebene Technik, um Pakete blockweise in den Grafikkartenspeicher zu kopieren. Ebenso wird eine FIFO-Warteschlange darüber geführt, welche Blöcke in der GPU in Bearbeitung sind.

Sind die Berechnungen aller Pakete in einem Block fertig, wird dieser als ganzes wieder in den Arbeitsspeicher zurück kopiert. Anschließend wird überprüft ob es einen Block in der Warteschlange gibt, der vorher eingefügt wurde, und somit auf dessen Verarbeitung gewartet werden muss. Ist die Reihenfolge korrekt, können wieder einzelne Pakete aus dem Block für weitere Verarbeitung auf der CPU freigegeben werden. In Abbildung 5 sind die einzelnen Schritte dieser beiden Schnittstellen dargestellt.

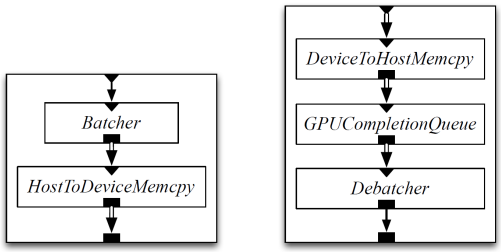


Abbildung 5: Snap GPU-Schnittstellen (Quelle: [11])

Zur Vermeidung unnötiger Kopiervorgänge lassen sich in Snap sogenannte Regions of Interest (ROIs) innerhalb der Netzwerkpakete spezifizieren. Mit diesem Trick muss nur das zur Verarbeitung relevante Teil der Header kopiert werden. Sollten diese Bereiche durch mögliche Optionen im Header an dynamischen Positionen liegen, muss diese bereits vor dem Kopiervorgang bestimmt werden. Im Gegensatz zu Zero-Copy Mechanismen können so trotz Kopiervorgängen die Datenmengen klein gehalten und der Durchsatz gesteigert werden.
Durch die pipelineartige Verarbeitung entlang des Graphen entsteht ein weiteres Problem: was geschieht mit Paketen in einem Block, die unterschiedlich bearbeitet werden müssen? Die Autoren von Snap [11]haben dazu Messungen von zwei Konzepten gemacht. Die eine Möglichkeit bestand in Aufsplittung des Pfades auf der GPU. Durch die verwendete SIMD-Architektur von GPUs reduziert sich durch das Aufsplitten die Leistung, da nicht mehr alle Daten die gleichen Verarbeitungsschritte durchlaufen. Die zweite Möglichkeit ist das Klassifizieren von Paketen und Anhängen von Metadaten. Danach durchlaufen alle Pakete jedes Element, wobei nur die zu den Metadaten passenden bearbeitet werden. Die Klassifizierung kann dabei sowohl in dem von der GPU bearbeiteten Bereich als auch in dem der CPU stattfinden. Abbildung 6 verdeutlicht die Klassifizierung und zeigt wie eine GPU-Pipeline genutzt werden kann um divergente Pfade zu ermöglichen. Die Messungen ergaben, dass die Klassifizierung höhere Leistung erzielt, obwohl jedes Paket alle Element durchläuft.

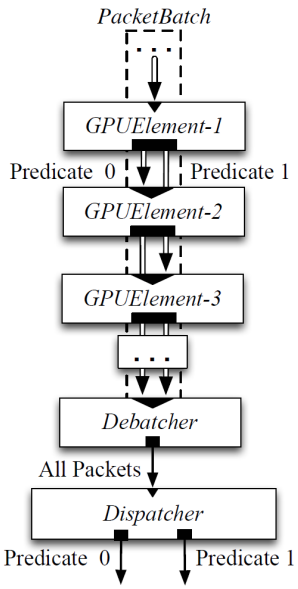


Abbildung 6: Konvergente Pfade der GPU-Pipeline (Quelle: [11])

4.2 PacketShader

PacketShader [3] ist im Gegensatz zu Snap, welches eine Erweiterung zu einem weit verbreiteten Framework darstellt, ein eigenständiges. Es ermöglicht ebenfalls wie Snap eine Programmierung des Routers im user-Level. Hierfür entwickelten die Autoren drei wichtige Komponenten: eine im Kernel integrierte Schnittstelle zur Kommunikation mit den NICs, eine auf Multithreading ausgelegte Steuerungslogik und den Verarbeitungsprozess.

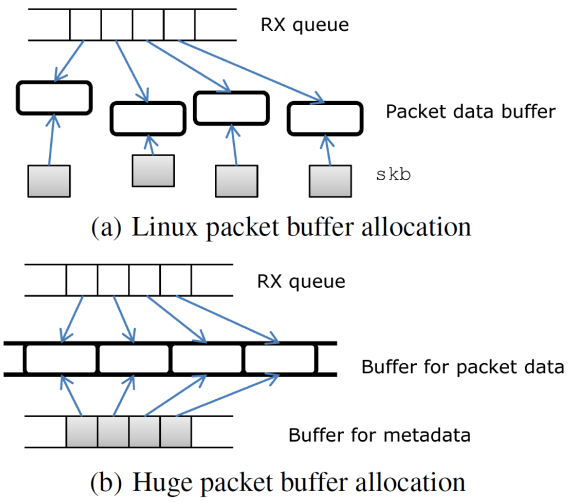


Abbildung 7: Pufferzuweisung von PacketShader (Quelle: [3])

Die Netzwerk-Schnittstelle wurde gegenüber dem Linux Kernel grundlegend verändert. Während Linux für jedes empfangene Paket ein eigenen Pufferbereich samt Metadaten anlegt, werden nun zwei großer Ringpuffer mit festen Zellengrößen für Daten und Metadaten bereitgestellt. Jedem Element im Empfangspuffer wird nun eine Position innerhalb des Ringes zugewiesen. Durch diese Veränderung lässt sich

doi: 10.2313/NET-2014-08-1_05

die Fragmentierung der Speicherbereiche effizient umgehen, da Pakete auch im Speicher bündig aneinander liegen. Durch die vertikale Streuung oben, sowie die strikte Anordnung der Puffer unten, versucht die Abbildung 7 diese Anpassung grafisch zu veranschaulichen. Da die Pakete nicht mehr durch den Netzwerk-Stack von Linux müssen, konnte zusätzlich die Größe der Metadaten von 208 auf 8 Bytes reduziert werden. Somit werden nur noch die für einen Software-Router relevanten Daten gespeichert.

Zur Kommunikation zwischen Host bzw. CPU und GPU verwendet PacketShader spezielle Threads. Messungen belegten eine Verschlechterung der Leistung, wenn zu viele Threads mit der GPU kommunizieren wollen. Gelöst wurde dieses Problem durch Unterteilung der Threads in 'Master' und 'Worker'. Ein Master-Thread ist ausschließlich zu Kommunikation zwischen einem CPU-Kern und der GPU dar. Ein Worker-Thread regelt hingegen die Pakete und den damit verbundenen I/O. Dabei dient der Master-Thread als Vermittler beim Zugriff auf die GPU. Um Cache-Misses gering zu halten, arbeitet jeder Thread nur auf einem ihm zugewiesenen CPU-Kern.

Die Verarbeitung ist wiederum die höchst gelegene Steuerinstanz von PacketShader und ist wiederum in drei Abschnitte gegliedert:

- **pre-shader:** In diesem Schritt wird ein Block aus einem Warteschlangenpuffer geladen, klassifiziert die enthaltenen Pakete und erstellt eine neue Datenstruktur um sie auf die GPU zu kopieren. (Abbildung 8 links)
- **shader:** Der Master-Thread kopiert die die Daten auf die GPU, startet dort ihre Ausführung und überträgt die verarbeiteten Paketblock wieder auf die entsprechende Warteschleife des Worker-Threads. (Abbildung 8 mitte)
- **post-shader:** Im letzten Schritt können noch weitere Operationen durchgeführt werden, bevor der Block wieder in die einzelnen Pakete aufgeteilt wird und an die entsprechenden NICs gesendet werden. (Abbildung 8 rechts)

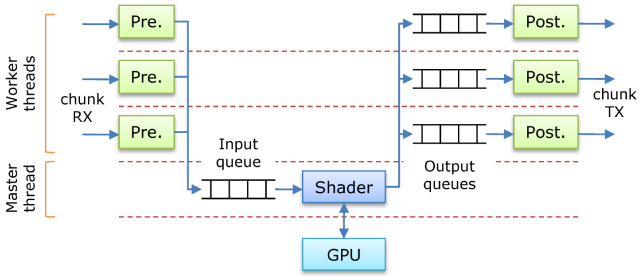


Abbildung 8: Ablauf der Verarbeitung durch die Master- bzw Worker-Threads (Quelle: [3])

Abbildung 9 verdeutlicht den Zusammenhang aller genannten Bestandteile. Für weitere Optimierungen nutzen die Autoren ein Feature namens *concurrent copy and execution* der verwendeten NVIDIA Karten, das gleichzeitiges Kopieren und Verarbeiten erlaubt. Dadurch können die Recheneinheiten auf der GPU ohne zu Pausieren neue Daten verarbeiten.

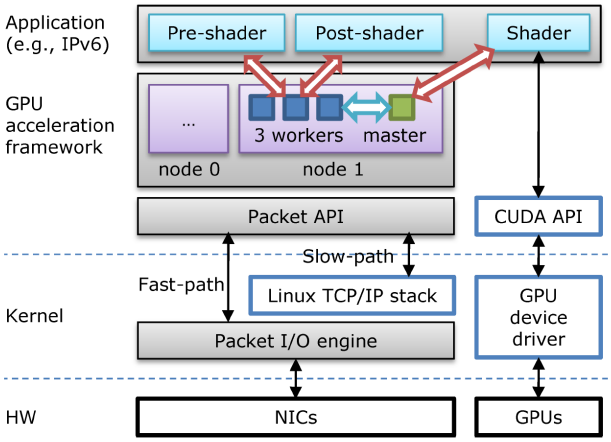


Abbildung 9: Interne Struktur von PaketShader (Quelle: [3])

4.3 Vergleich

Beide Ansätze unterscheiden sich in der Art ihrer Implementation deutlich von einander. Während die Autoren von Snap auf die Erweiterung existierender Technik setzen, wurde bei PacketShader ein eigens Konzept entwickelt. Dabei lässt sich gut beobachten wie beide Umsetzungen die gleichen grundlegenden Techniken verwenden. Der große Vorteil von Snap ist die einfache Integration in bestehende Systeme auf Basis von Click.
Vergleicht man die errichten Leistungen erhält man ein sehr ähnliches Bild. In beiden Fällen erreichen schlichte Router Konfiguration eine Leistung von etwa 40 GBit/s. Im Gegensatz zu früheren Projekten stellt das eine Verbesserung dar und beweist die Skalierbarkeit von Software-Routern.

5. FAZIT

Zusammenfassend kann man damit belegen, dass sich Grafikprozessoren auch in dem Bereich der Netzwerkverarbeitung effektiv Einsätzen lassen. Als Entwickler entsprechender Framworks gibt es einige Aspekte die es zu beachten gilt. Es müssen Lösungen gefunden werden, die eine genaue Kenntnis der verwendeten Hardware aber ebenso gleichzeitig Wissen über Softwareoptimierungen verlangen. So spielt ein gutes Speichermanagement für einen Software-Router eine erhebliche Rolle. Dennoch können die einzelnen Techniken auf verschiedenen Systemen unterschiedlich starke Auswirkungen haben. Ebenso sind die Laufzeiteinstellungen, z.B. die Blockgröße für die GPU, die Anzahl der Threads etc, stark von der verwendeten Hardware abhängig.
In den häufigsten Einsatzgebieten aktueller Software-Router, wie zum Beispiel der DSL-Router im eigenen Haus, reicht oft geringer Datendurchsatz. Aus diesem Grund beschränkt sich der Einsatz von GPU-beschleunigten Implementierungen hauptsächlich auf akademische Forschungsarbeiten. Dort beschränkt sich der Einsatz jedoch nicht nur auf Router Funktionalitäten sondern auch auf Firewalls oder Intrusion Detection Systeme [14].

In Zukunft könnte die Verwendung von Grafikprozessoren noch um einiges leichter werden. Durch den großen Erfolg GPU-basierter Softwarelösungen streben die Hardwarehersteller immer bessere Nutzbarkeit für Entwickler an. Ein

doi: 10.2313/NET-2014-08-1_05

großer Schritt für die nähere Zukunft wird die Einführung leistungsstärkerer All-In-One Chips, die bisherige dedizierte Zusatzkarten überflüssig machen könnten. AMD hat bereits mit ihrem HSA-Programm erste Grundsteine gelegt und auch Intel ist mittlerweile auf einem ähnlichen Weg [12] und integriert eine stetig schnellere Grafikeinheit in ihre Prozessoren.

6. LITERATUR

[1] M. Dobrescu, N. Egi, K. Argyraki, B.-G. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, and S. Ratnasamy. RouteBricks: Exploiting Parallelism to Scale Software Routers. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, 2009.

[2] S. Han, K. Jang, K. Park, and S. Moon. Building a single-box 100 gbps software router. In *In IEEE Workshop on Local and Metropolitan Area Networks*, 2010.

[3] S. Han, K. Jang, K. Park, and S. Moon. Packetshader: A gpu-accelerated software router. *SIGCOMM Comput. Commun. Rev.*, 40(4):195–206, Aug. 2010.

[4] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Trans. Comput. Syst.*, 18(3):263–297, Aug. 2000.

[5] G. Kyriazis. Heterogeneous System Architecture: A Technical Review. Website, (21.03.2014), Mar. 2012. `http://developer.amd.com/wordpress/media/2012/10/hsa10.pdf`.

[6] NVIDIA Corporation. NVIDIA's Next Generation CUDA Compute Architecture: Kepler GK110. Website, (23.03.2014), 2012. `http://www.nvidia.com/content/PDF/kepler/NVIDIA-Kepler-GK110-Architecture-Whitepaper.pdf`.

[7] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. Lefohn, and T. J. Purcell. A survey of general-purpose computation on graphics hardware. In *Eurographics 2005, State of the Art Reports*, pages 21–51, Aug. 2005.

[8] L. Rizzo. netmap: A novel framework for fast packet i/o. In *Presented as part of the 2012 USENIX Annual Technical Conference (USENIX ATC 12)*, pages 101–112, Boston, MA, 2012. USENIX.

[9] R. Sietmann. Der fünfte Weg: Die Zukunft von IPTV, Internet-TV und die Netzneutralität. Website, (22.03.2014), 2011. `http://www.heise.de/ct/artikel/Der-fuenfte-Weg-1167899.html`.

[10] R. Smith. NVIDIA Announces CUDA 6: Unified Memory for CUDA. Website, (25.03.2014), 2013. `http://www.anandtech.com/show/7515/nvidia-announces-cuda-6-unified-memory-for-cuda`.

[11] W. Sun and R. Ricci. Fast and flexible: Parallel packet processing with gpus and click. In *ANCS*, pages 25–35. IEEE, 2013.

[12] Tom Piazza. Processor Graphics. Website, (22.05.2014), 2012. `http://www.highperformancegraphics.org/previous/www_2012/media/Hot3D/HPG2012_Hot3D_Intel.pdf`.

[13] J. Tompson and K. Schlachter. An introduction to the opencl programming model. 2012.

[14] G. Vasiliadis, S. Antonatos, M. Polychronakis, E. P. Markatos, and S. Ioannidis. Gnort: High performance network intrusion detection using graphics processors. In *Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection*, RAID '08, pages 116–134, Berlin, Heidelberg, 2008. Springer-Verlag.

A Survey of Trends in Fast Packet Processing

Konstantina Tsiamoura
Betreuer: Florian Wohlfart, Daniel G. Raumer
Seminar Future Internet SS2014
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: k.tsiamoura@tum.de

ABSTRACT
Network interfaces of 10 Gbit/s are used more and more. Despite their high performance, packet processing rates are not correspondingly higher. This happens due to the architecture of the network stacks and the design of device drivers. There are restrictive factors and overheads which need to be overcome, so as to enable fast packet processing with commodity hardware. In this paper, techniques towards this direction are discussed and existing frameworks which exploit combinations of them are presented and compared. The frameworks mentioned are divided into two main categories, depending on whether they involve GPU for the packet processing or not.

Keywords
fast packet processing, GPU packet processing, zero-copy techniques

1. INTRODUCTION
Network interfaces have increased their performance and 10 Gbit/s rates tend to be nowadays commonplace. However, the increased performance of these network interfaces cannot by itself guarantee packet processing at high speeds. There are overheads imposed by the network stacks' architectural design which was meant to be used with general-purpose hardware. It emphasizes on compatibility and overlooks performance [3]. Packet processing can become more efficient by alternating the network stack and without necessitating more powerful hardware [15]. This fact has led to surveys concerning how general-purpose hardware and fast network interfaces can be effective regarding packet processing. 10 Gbit/s entail a considerable high amount of packets per second [16], which demands intensive CPU usage, and the support of packet processing at high rates without packet losses.

In order to understand which techniques would be beneficial to making general-purpose hardware suitable for high speed network applications, an understanding of its current restrictive parameters is required. These factors refer to hardware resources limitations and the way that packet processing is organized based on the network stack. Afterwards, the presentation of widely suggested techniques which target to avoid or decrease the problems resulting from these factors is meaningful.

Two of the proposed techniques hold a special position. The first one is zero-copy technique, which eliminates the overhead of the data copy between kernel and user-space. It is implemented by the majority of the existing frameworks for fast packet processing. The second technique is more recent and emerging and refers to the exploitation of GPU to perform parts of the packet processing.

Frameworks aiming to enable fast packet processing are presented as well as the way the proposed techniques can be put into practice.

The rest of the paper is organized as follows: In Section 2, the state-of-the art New API (NAPI) and the Click modular router are presented. Section 3 provides an overview of the factors that restrict the performance of general-purpose hardware towards packet processing in combination with recommended solutions to them. Section 4 is about zero-copy technique and frameworks that make use of them in order to enable a faster packet processing. In Section 5, Netslice is presented; a framework that uses neither zero-copy technique, nor GPU for the packet processing. Frameworks that exploit GPU processing are presented in Section 6. Finally, in Section 7 a comparison between the frameworks, in terms of the techniques they use, takes place.

2. STATE OF THE ART

In this section the state-of-the art in packet processing is presented, by the description of the Linux IP stack and its NAPI, and the Click Modular Router. This section provides also some essential background for network stacks and the packet flow through them during packet processing.

2.1 LINUX NETWORK STACK AND THE NEW API (NAPI)
The Linux network stack is introduced by the description of the packet flow which takes place when a packet arrives. This packet flow is represented in figure 1.

The packet arrives at one of the circular receiving queues (RX) of the Network Interface Controller (NIC), which are also called rings. There, the packet is stored in a data structure, the receiver descriptor, which enables the copying of data between the NIC and the main memory. The data transfer is achieved via the Direct Memory Access (DMA) mechanism, which copies the data to the DMA-able memory region without the CPU. At this point a mechanism is needed to inform the system that a packet has been received so as to perform the data transfer between the

doi: 10.2313/NET-2014-08-1_06

DMA-able memory region and the packet buffer that Linux kernel allocates for each packet. For that purpose, in Linux network stack, an interrupt-based mechanism is used and an interrupt is raised at every packet's arrival. Then the packet has to be copied from the kernel to user-space [3].

The interrupt mechanism is not suitable for packets arrival at high rates, since it causes the receive or interrupt livelock phenomenon [2]. This phenomenon results in reducing the system's performance because all resources are devoted to interrupts handling and not to real packet processing related tasks [12, 5].

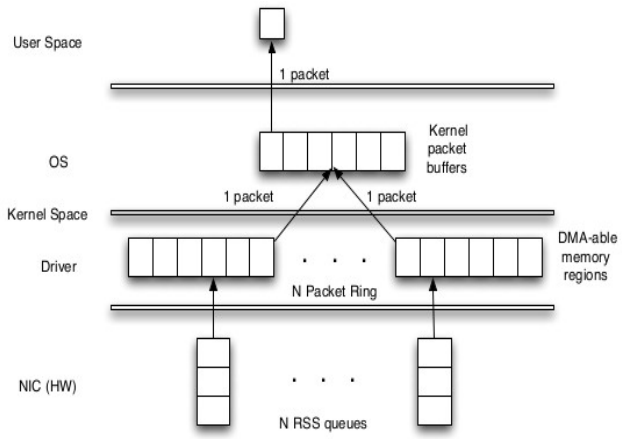


Figure 1: Linux Network Stack [3]

NAPI was introduced in the initial Linux network stack in order to overcome the receive livelock problem, by using a polling mechanism in addition to the interrupts, to exploit their benefits at high and low speeds of packet arrival respectively. NAPI can interchange these mechanisms dynamically. Interrupts are only enabled for the first packet of a batch. Then a polling mechanism gets periodically enabled, to inspect the devices that have received packets that want to forward to the network stack [19]. These packets are temporarily stored to the DMA-able memory region, waiting for the CPU to become available. The interrupts mechanism can be activated again on the condition that there are no more available packets in the DMA-able memory [3]. This hybrid mechanism ensures that when the traffic is low, there will be no increased latency due to the interrupts. In addition, when there is packet arrival at high rates the polling mechanism ensures: 1) reduced cache misses, 2) reduced I/O overhead, and 3) better CPU throughput [20].

2.2 Click Modular Router

Click modular router constitutes a framework for creating modifiable and flexible routers. Click, which can be used instead of the Linux IP stack [11], can be executed in kernel as well as in the user-space. Similar to NAPI, Click uses both interrupts and polling mechanism [1].

With Click, it is feasible to develop most of the packet processing software by putting together elements in a pipeline structure [23]. Click's simple architecture is enhanced due to the pull processing and the flow-based router context [13].

The components of the Click are called elements. They are individual units related to routing and forwarding processing, for instance filters and queues [11], which can incrementally implement a router configuration. A router configuration is defined as a graph, where nodes represent the elements, and edges represent the direction of the packets as they are moving through the elements via their input and output ports. The ports are responsible for establishing the connection between the elements.

The pull processing, as well as the push processing, are two functions that determine the way in which the elements communicate with each other and which entity causes a packet to travel from one element to another. According to the push processing, the sending element delivers a packet to a receiving element, while in the pull processing, it is the receiving element which triggers the move of a packet by asking it from the corresponding sending one. Sending and receiving elements are also referred to as upstream element, which are found at the top of the pipeline, and downstream element, which are found at the bottom of the pipeline structure respectively [23]. An example of a Click router configuration is depicted in figure 2. The black and the white ports are used to represent push and pull ports respectively. Queues are distinct entities in Click and are represented by a distinct element, the *Queue* element. Such an element is the second element of the router configuration.

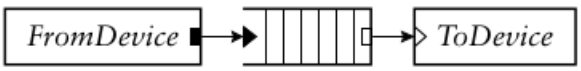


Figure 2: Example of a Click router configuration [13]

The second important trait of the Click Modular Router is the flow-based router context. It is information available to an element, concerning the path that a packet needs to follow. With this information, the element can identify all the other elements which constitute the flow of the packet and not only these with which it has an immediate link.

3. FAST PACKET PROCESSING TECHNIQUES

Communication links have reached and exceeded the rate of 10 Gbit/s. This fact raises the question, how can general-purpose systems support the huge number of packets that 10 Gbit/s link entail [16]. Therefore, solutions that can achieve fast packet processing are needed.

3.1 Fast Packet Processing Impediments

In [20] the following parameters are considered to affect the performance of packet processing applications: 1) CPU speed and inadequate utilization, 2) interrupts overhead, 3) limited bus bandwidth in comparison to a fast processing unit, 4) memory latency, 5) I/O latency.

doi: 10.2313/NET-2014-08-1_06

In [3], there is a more detailed description of the factors related to the first parameter mentioned above, meaning the CPU, that restrict high speed packet processing. More specifically these overheads originate from: 1) performing memory allocation and deallocation for each packet (per-packet memory allocation), 2) overhead of copying data between kernel and user-space, 3) expensive cache misses, 4) per-packet system calls, because of the CPU-intensive context switch between kernel and user-space, and 5) the transformation of the parallelized processing of packets by the queues of multi-queue NICs to a serialized one. This happens because all packets have to converge to one single point, thus creating a bottleneck.

3.2 Techniques against Fast Packet Processing Impediments

In [3], techniques that have been proposed as a countermeasure against the limitations and overheads of packet processing have been presented. Table 1 associates the problems discussed in Section 3.1 with their proposed solution. These techniques are widely adopted by the frameworks that are going to be presented in the following sections.

Table 1. Solutions for the CPU-related problems

Problem	Solution
Per-packet memory allocation	Memory pre-allocation
Copy of data between kernel and user-space	Zero-copy techniques
Cache misses	Memory affinity, prefetching
System calls	Batch packet processing to reduce system calls
Not parallel packet processing to the whole network stack	Direct paths from NICs to user-space that can support parallelism

4. FRAMEWORKS SUPPORTING ZERO-COPY TECHNIQUES

4.1 Zero-copy technique

As it has been already mentioned, one significant overhead of the packet processing is caused because of the data copies between kernel and user-space. To avoid data copies, a zero-copy technique needs to be applied. The zero-copy technique actually refers to a collection of techniques that reduce the copies between either kernel and devices or kernel and user-space [22].

4.2 Netmap

Netmap [16] is a framework that performs packet processing at high speeds by reducing the avoidable costs that slow the processing down. It enables the applications to gain fast access to the packets. Netmap is independent of specific devices and hardware, and is built upon existing operating system characteristics and applications [15].

In the figure 3, the Netmap architecture is presented. In particular the data structures that Netmap incorporates in its

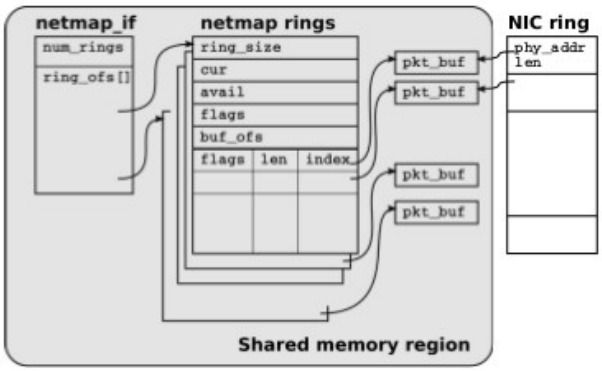


Figure 3: Netmap Data Structure [14]

architecture are depicted, namely packet-buffers, netmap rings and netmap_if. These data structures are all located to a shared memory region which can be accessed both by kernel and user-space applications, so as to avoid copying data. The shared memory region is mapped to all user processes address space via the system call *mmap()* [17]. It is important that due to the contents of the shared memory region, an application running on Netmap cannot lead to a kernel crash.

Packet-buffers: The packet buffers of Netmap have a predetermined size and are preallocated. This reduces the cost that per-packet allocations and deallocations entail. The size of the buffer is sufficient for storing packets without fragmentation. The packet buffers are used by both netmap and NICs' rings.

Netmap rings: They are circular queues that contain metadata related to the buffer, similar to NICs rings. They contain information like the count of data entries, *slots*, that the ring can store, the count of available buffers and an array of slots. Netmap decouples the NIC rings from the network stack, and it allows user-space applications to communicate with them separately and for different reasons through the netmap rings. More specifically, Netmap exploits the speed with which a NIC can transfer packets between the network and the memory. Synchronization can be achieved through operating system functions like *select()* and *poll()*.

Netmap_if: The netmap_if data structure includes information related to the interface, like how many rings there are.

The presented data structures offer the following advantages: 1) reduced per-packet cost, 2) efficient communication between interfaces and between NIC and network stack and 3) potential use of multi-queue NICs.

4.3 RF_Ring

PF_Ring [14] is a framework that provides fast packet capturing and processing. PF_Ring implements zero-copy by avoiding data copy between the kernel and the user-space. It achieves that with the use of packet buffers that are found in a memory region common to the kernel and the users' applications [16]. The

doi: 10.2313/NET-2014-08-1_06

packet buffers are preallocated. Due to this packet buffer schema, it avoids the cost of per-packet memory allocation and deallocation. The basic elements that compose PF_Ring's architecture are:

PF_Ring kernel module: It is responsible for copying the packets to the PF_Ring circular queues.

PF_Ring user-space library: It is through this library that PF_Ring kernel module is exposed to user-space applications.

PF_Ring aware drivers: Although PF_Ring is NIC independent, which means that it is not based on specific NICs, it can be used with specialized drivers too and gain more in terms of performance.

4.3.1 PF_Ring DNA (Direct NIC Access)

The PF_Ring can be used with a specialized type of device driver, namely DNA, for achieving even faster processing without the intervention of CPU and the use of system calls [18]. PF_Ring provides a mapping between the NIC memory and the user-space memory and permits the explicit communication between applications and NICs. This can be seen in figure 4. The packets are transferred between the NIC and the user-space, without the intervention of either Linux kernel or PF_Ring module. In this case, there is only the data copying performed by the NIC Network Process Unit (NPU) via DMA, while the copy from the kernel packet buffer is omitted. This is referred to, as full zero-copy [3].

Besides the high speed performance that can be achieved by the PF_Ring DNA, this approach has two important disadvantages. First, it entails the risk of misusing memory addresses via the NIC's DMA engine and thus leading to a possible system crash [16]. Second, its use is limited to one application at a time. [14]

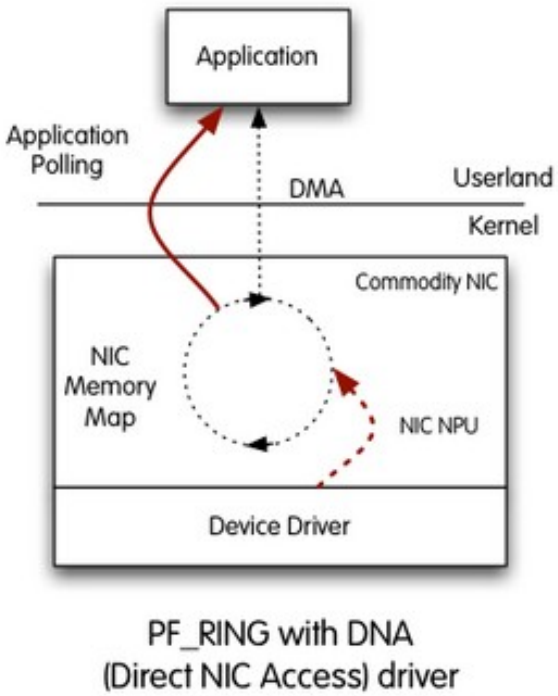


Figure 4: PF_Ring DNA Architecture [6]

4.4 INTEL Data Plane Development Kit

The Intel DPDK [7] is a framework that provides fast-packet processing to applications related to the data plane part, meaning to those applications that are responsible for forwarding packets to their applications [21].

The framework supports two models for processing the packets. It can implement either a run to completion model or a pipeline one. In the run to completion model, every processing unit is allocated to one packet, which it completely processes. All the processing units execute a common application code. Contrary to this, the processing units of the pipeline execute different application code, since they perform a distinct task of the packet processing. A packet is not entirely processed by a single thread, and it has to be transferred through every processing unit of the pipeline.

4.4.1 Intel DPDK Architecture

Environment Abstraction Layer (EAL): The framework provides libraries for particular environments. It also implements the Environment Abstraction Layer in order not to expose details concerning each environment to the libraries and to the data plane applications that will use them. Through the interface provided by the EAL, applications can gain access to hardware related resources.

More specifically, the EAL 1) allocates memory, 2) devotes cores to execution units, 3) communicates with the PCI bus, and 4) inspects the existence of interrupts.

doi: 10.2313/NET-2014-08-1_06

Core Components: The core components define the libraries offered by Intel DPDK for achieving fast packet processing.

Table 2. Intel DPDK's core components

library	Description
Memory Manager	Services for memory allocation
Ring Manager	Defines the Ring data structure for storing the packets
Memory Pool Manager	Allocates portions of memory of specific length and stores object by using rings
Network Packet Buffer	Manages packet buffers
Timer Manager	Services for scheduling functions

4.4.2 Intel DPDK Fast Packet Processing Techniques

The Intel DPDK manages to perform packet processing at high speeds due to the following characteristics:

- The ring data structure, which is implemented as queue, provides faster access and storage than a linked list. Besides, it diminishes the time needed for time-consuming bulk operations.

- It uses pre-allocated buffers. In fact, there is only one buffer for keeping both metadata information and packet data and so there is only one memory allocation per-packet.

- It enables the cores to have their own cache memory and limits their accesses to the shared ring of a memory pool, which can be CPU-intensive.

- It reduces interrupts overhead with the use of the Poll Mode Drivers.

5. NETSLICE

In this section, NetSlice [10] is presented. Contrary to the solutions of fast packet processing presented in the previous section, NetSlice does not take advantage of zero-copy techniques.

NetSlice is an operating system abstraction that attempts to provide high-speed packet processing and runs in the user-space. More specifically, it aims to reconcile the benefits of the packet processing applications that run in the user-space, i.e fault isolation and programmability, with the performance of 10 Gbit/s of current NICs. NetSlice is based on the high coupling of software and hardware components that are related to packet processing. Moreover, it allows these components to be managed by the applications and reduces the contention between CPUs.

NetSlice is based on spatial instead of temporal partitioning of the software and hardware components that are relevant to packet processing, i.e of the memory, the CPU cores and the NICs. By performing such partitioning, NetSlice minimizes the contention of the shared resources.

The elementary notion of the NetSlice is the execution context, named *NetSlice*. In fact, there is an array of such execution contexts, in order to provide high parallel execution, as well as keeping contention rates low.

In figure 5, an array of NetSlices is depicted. As mentioned above, NetSlice takes advantage of multi-core and multi-queue NICs. In order to support parallel execution at the multiple cores, the multi-queue NICs maintain more than one queue for transmission and reception. The NetSlice execution context contains implicitly partitioned resources and the CPU cores and NICs which are also referred to as explicitly partitioned resources. More specifically, each NetSlice must have at least two CPU cores. The CPU cores of a NetSlice execution context are classified into *k-peer*, and *u-peer*, where the in-kernel and user-mode tasks are executed correspondingly. Only one of these CPU cores, which has been specified to receive the interrupts of the queues of the context, is the k-peer CPU and is assigned to execute the in-kernel network stack. The user-mode task is executed in parallel on the u-peer CPU. The k-peer and the u-peer CPU cores of an execution context are defined as tandem CPU cores, and it is the number of tandem CPU cores that determines the count of the NetSlices.

NetSlice also determines the path that packets have to follow from NICs to applications and vice versa. Packets are slightly processed on the k-peer CPU core, and then they are directed to the user-space application, where they are processed in a pipeline.

NetSlice is based on the conventional socket API. More specifically, it uses the operations write, read and poll for the distinct data flows of the different NetSlices. NetSlice extends the API via the *ioctl* mechanism. One difference of the NetSlice extended API is the batched system calls that it can support. Due to the batching, NetSlice achieves a reduced number of system calls, and thus minimizes the delays of system calls. Batching results also in a reduction of the overheads that per packet processing poses.

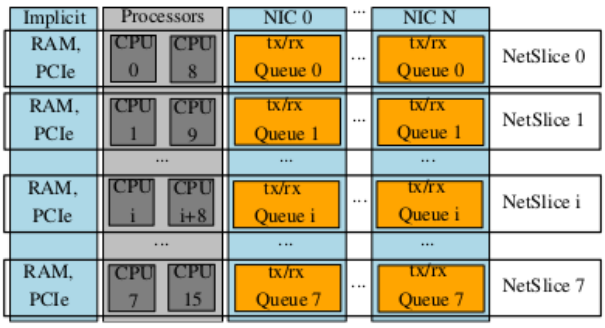


Figure 5: Array of NetSlices [10]

NetSlice does not exploit the advantages of zero-copy techniques, and it does copy the packets from kernel to the user-space. It is estimated that zero-copy technique could indeed further

doi: 10.2313/NET-2014-08-1_06

ameliorate the performance of NetSlice. However, it is not implemented in the framework, since it would restrict portability.

6. PACKET PROCESSING WITH GPUs

6.1 GPU

The architecture of the Graphical Processing Units with the hundreds or thousands of cores, have resulted in their use in other applications too, except their primary use in graphics rendering [8, 4]. Highly parallel applications can take advantage of the ample computation cycles and the ample memory bandwidth of GPUs, higher than that of CPUs, which enables the access to different data sets [8]. Among the applications that can be benefited from the highly parallel computing of GPUs are applications related to packet processing tasks.

6.1.1 GPU Overview

Due to the numerous cores that they possess, modern GPUs have gained a massively parallel processing power [23]. A GPU has a device memory, which is explicitly accessed by the GPU. Implicit access of the host memory can be achieved through DMA or through the Peripheral Component Interconnect Express (PCIe) bus. The PCIe bus is also necessary for the communication between GPU and the corresponding host CPU. GPUs are based on a SIMT (Single Instruction, Multiple Thread) execution model. According to this, many cores have a common program counter and threads are executed in parallel.

6.1.2 GPU Benefits in packet processing

GPUs provide high thread-level parallelism, while a CPU is organized in a way that increases the instruction-level parallelism. As a consequence, a GPU, contrary to a CPU achieves to minimize the memory access latency by executing a considerable number of threads.
The significant memory bandwidth of the GPU, can be used for memory-intensive tasks. This can be very beneficial particularly in cases where the CPU' s memory bandwidth is used for packet I/O. Similarly, the processing power of the GPU can be used for computing-intensive operations which run on software routers and thus overcome the bottleneck posed by the CPU.

6.2 PacketShader

PacketShader [4] constitutes a user-space framework for fast-packet processing that takes advantage of the GPU characteristics. What increases the speed of the PacketShader is, except the GPU computing power, the optimized packet I/O engine which achieves the performance of 10Gbit/s due to pipelining and batching.

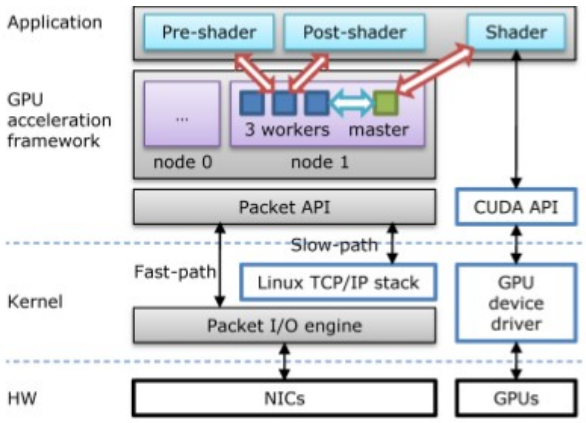


Figure 6: PacketShader Architecture [4]

6.2.1 PacketShader Architecture

The architecture of the PacketShader is depicted in the figure 6. Its more important components are analyzed below:

Packet I/O Engine: PacketShader does not use the per-packet buffer allocation scheme; it develops a new one, namely the huge packet buffer. According to the huge packet buffer scheme, the device drivers allocate one huge buffer for the metadata and another one for the real packet data. This optimized buffer allocation scheme offers two advantages. First, it does not result in CPU overhead as the per-packet buffer allocation does. Second, it reduces per-packet DMA mapping.

GPU Acceleration framework: The nodes in the GPU acceleration framework are NUMA (Non-uniform memory access) nodes and are used to increase parallelization. According to the NUMA model, the time needed to access the memory, is related to the position of the physical memory. The nodes contain CPUs with four cores. Three cores are allocated for worker threads and one for master thread.

Packet processing application: The packet processing application works with batches of packets instead of individual packets and is divided into three tasks, which are performed by the methods Pre-Shader, Post-Shader and Shader.

The Pre-Shader method is executed by a worker thread. It is responsible for packet classification and for forwarding data to the master thread. While waiting for the master thread to finish with the shading, the worker thread is processing the pre-shading for the next batch of packets.

Then the Shader method is performed. The master thread copies the data between the GPU device memory and the host memory, and launches the kernel, i.e the code that is executed by the GPU. The data copying and the execution of GPU kernel run concurrently in order to increase GPU performance. This technique is known as *concurrent copy and execution*. Afterwards, it returns the data to the worker thread where the Post-Shader method takes place. Another technique used in this step, for better performance, is the gather/scatter, which enables

doi: 10.2313/NET-2014-08-1_06

the system to process multiple batches of packets. The master thread gathers data from the worker threads and after the shading processing, scatters them in the corresponding worker threads.

During the post-shading, the packets can get modified, duplicated or discarded and then they are ready for transmission.

6.3 Snap

Snap [23] is another framework that enables fast packet processing due to both the processing power of GPU and the modular and flexible architecture of the Click modular router.

6.3.1 Snap Architecture

Snap exploits the extended parallelism that GPU offers. In the Snap framework, not all elements are offloaded to GPUs. Only elements that result in bottlenecks are implemented in GPUs. The other elements are still supported by the CPU as in the Click architecture. Besides, not all tasks of packet processing are suitable for highly parallel processing and therefore GPU is not appropriate for them.

Snap supports two types of elements. The first is the serial element and refers to Click's conventional elements, meaning those elements that fit in Snap's architecture without having undergone any change. The second type of elements is the GPU-based parallel element that Snap introduces in order to execute them in the GPU. This new type has, except the GPU code, a CPU part, which is responsible for receiving batches of packets and launching the kernel. For the communication of these elements, which process single packets and batches of packets respectively, a Batcher and a Debatcher element are needed. These elements act as adapters. The Batcher gathers individual packets from a serial element and inputs them in form of batches in a parallel element. Its opposite element, the Debatcher, receives batches of packets from a parallel element and transfers the packets separately to a serial element. By attributing the adapter functionality to specific elements, Snap allows the developer to monitor batching and offloading.

Batching is necessary for achieving high performance through GPU processing. Batching in Snap, as well as in PacketShader, differs from batching in Click, in that the latter refers to batches of elements that run on one individual packet, while Snap and PacketShader's batching signifies that a batch of packets is processed by the same element.

In order to be able to support this kind of batching, Snap introduces the functions *bpull()* and *bpush()* that perform the functionality of pull() and push() methods of Click for batches of packets. These modified functions can be performed by GPU-based parallel elements.

Similarly to the huge buffer allocation scheme that PachetShader uses, Snap uses huge buffers too for the batches of packets. In this way, it reduces per-packet DMA mapping.

As has been mentioned, GPU cores communicate only directly with the device memory. Therefore, copying packets between GPU and host memory through the PCIe bus is unavoidable. Snap diminishes the times that such a copying is necessary and consequently the memory overhead by implementing two types of elements, the *HostToDeviceMemcpy* and the *DeviceToHostMemcpy* element and putting them at the edges of a sequence of parallel elements.

Snap manages to overcome the problem of packet reordering which has been observed in parallel packet processing. This happens because it requires that all threads of a parallel element that processes a batch of packets must have finished before the batch of packets can be processed by the next parallel element. For avoiding a potential packet reordering because of the asynchronous GPU scheduling, a GPUCompletionQueue element is implemented and placed between the DeviceToHostMemcpy and the Debatcher.

7. COMPARISON OF THE FRAMEWORKS

In this section, the solutions presented above are compared. More specifically a comparative table showing which of the techniques discussed in the previous sections are implemented by each framework, is presented.

	Netmap	PF_Ring DNA	Intel DPDK	NetSlice	Snap	PacketShader
Memory preallocated, re-used	yes	yes	yes			yes
Parallel direct paths				yes		yes
Memory mapping	yes	yes	yes			yes
Zero-copy	yes	yes	yes			
Batch processing	yes			yes	yes	yes
CPU affinity	yes		yes	yes		yes
Memory affinity			yes	yes		yes
Aggressive prefetching						yes
GPU processing					yes	yes

Figure 7: Techniques per framework

8. CONCLUSION

Commodity network interface cards can achieve the speed of 10 Gbit/s. As a consequence, the need for performing packet processing at a corresponding rate with general-purpose hardware arose. In this paper, a survey of current techniques for high-speed packet processing was conducted. The survey was mostly focused on existing frameworks developed for serving fast packet processing. This survey described and compared several frameworks which provide fast packet processing. These frameworks actually manage to do so, by applying proposed techniques to the network stack and to the way that packet processing is performed. A newly proposed and emerging technique, that some recent frameworks have adopted, involves the incorporation of GPU for processing parts of packet processing.

9. REFERENCES

[1] Bianco, A., Birke, R., Bolognesi, D., Finochietto, J., Galante, G., Mellia, M., Prashant M., and Neri F. 2005.

Click vs. Linux: Two Efficient Open-Source IP Network Stacks for Software Routers. IEEE High Performance Switching and Routing Workshop, Hong Kong, May 2005.

[2] Deri, L. 2004. Improving Passive Packet Capture: Beyond Device Polling. Proceedings of the 4th International System Administration and Network Engineering Conference (SANE), 2004.

[3] Garcıa-Dorado, J., Mata, F., Ramos, J., Santiago del Rio, P., Moreno, V., and Aracil, J. 2013. High-Performance Network Traffic Processing Systems Using Commodity Hardware, in Biersack, E., Data Traffic Monitoring and Analysis, (Berlin: Springer, 2013).

[4] Han, S., Jang, K., Park, K., and Moon, S., Packetshader: a gpu-accelerated software router. ACM SIGCOMM Computer Communication Review 40, 4 (2010), 195–206.

[5] Indiresan, A., Mehra, A., and Shin, K. 1998. Receive Livelock Elimination via Intelligent Interface Backoff. TCL Technical Report (Michigan: University of Michigan, 1998).

[6] Ine, J. 2014. PF_RING User Guide, Version 5.6.2, Jan 2014, ntop.org

[7] Intel. 2014. Intel® Data Plane Development Kit (Intel® DPDK), Programmer's Guide.

[8] Jang, K., Han, S., Han, Se., Moon, S., and Park K. 2011. SSLShader: Cheap SSL Acceleration with Commodity Processors. USENIX NSDI, April 2011.

[9] Kohler, E., Morris, R., Chen, B., Jannotti, J., and Kaashoek, M. 2000. The Click modular router. ACM Trans. Comput.Syst., 18(3):263–297, Aug. 2000.

[10] Marian, T., Lee, K., and Weatherspoon, H. 2012. Netslices: Scalable Multi-Core Packet Processing in User-Space, ANCS'12, October 29–30, 2012, Austin.

[11] Meyer, T., Wohlfahrt, F., Raumer, D., Wolfinger, B., and Carle, G. 2013. Measurement and Simulation of High-Performance Packet Processing in Software Routers. 7th GI/ITG-Workshop MMBnet, Hamburg, September 2013.

[12] Mogul, J., and Ramakrishnan , K. 1997. Eliminating receive livelock in an interrupt-driven kernel. ACM Transactions on Computer Systems (TOCS) 15, 3 (1997), 217–252.

[13] Morris, R., Kohler, E., Jannotti, J., and Kaashoek, M. 1999. The Click modular router. Operating Systems Review 34(5):217–231, December 1999

[14] NTOP. http://www.ntop.org/products/pf_ring/dna

[15] Rizzo, L. 2012. Revisiting Network I/O APIs: The Netmap Framework. Communications of the ACM, vol. 55 no. 3, March 2012.

[16] Rizzo, L. 2012. Netmap: a novel framework for fast packet I/O. Proceedings of the 2012 USENIX conference on Annual Technical Conference, USENIX ATC'12, Berkeley, CA, 2012.

[17] Rizzo, L., Carbone, M., and Catalli, G. 2012. Transparent acceleration of software packet forwarding using netmap. INFOCOM'12, Orlando, FL, March 2012, http://info.iet.unipi.it/∼luigi/netmap/.

[18] Rizzo, L., Deri, L., and Cardigliano, A. 2012. 10 Gbit/s Line Rate PacketProcessing Using Commodity Hardware: Survey and new Proposals. Online: http://luca.ntop.org/10g.pdf (2012).

[19] Salim, J., Olsson, R., and Kuznetsov, A. 2001. Beyond Softnet. Proceedings of the 5th Annual Linux Showcase & Conference - USENIX Association, (Oakland,CA, 2001).

[20] Salim, J. 2005. When NAPI Comes to Town. Linux 2005 Conference.

[21] Schuchard, M., Vasserman, E., Mohaisen A., Kune, D., Hopper, N., and Kim, Y. 2011. Losing Control of the Internet: Using the Data Plane to Attack the Control Plane, ISOC Network & Distributed System Security Symposium (NDSS 2011).

[22] Song, J., and Alves-Foss, F. 2012. Performance Review of Zero Copy Techniques. International Journal of Computer Science and Security (IJCSS), Volume (6) : Issue (4) : 2012.

[23] Sun, W., and Ricci, R. 2013. Fast and Flexible: Parallel Packet Processing with GPUs and Click. Proceedings of the 9th ACM/IEEE symposium on Architectures for networking and communications systems, 2013.

Internet Science – High Reliability or When Things go wrong and as things change

Tilman Eberspächer
Betreuer: Heiko Niedermayer
Seminar Future Internet SS 2014
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
eberspae@in.tum.de

KURZFASSUNG

Fehler passieren jeden Tag und überall. Gründe hierfür sind menschliches Fehlverhalten, maschinelle Ausfälle, fehlerhafte Arbeitsabläufe, mangelnde Kommunikation und viele mehr. In seinem Buch „Friendly Fire“ analysiert Scott A. Snook welches Fehlverhalten und welche Rahmenbedingungen am 14. April 1994 zum Abschuss zweier Transporthubschrauber vom Typ UH-60 Black Hawk durch zwei U.S. Air Force F-15 Kampfflugzeuge im Luftraum über dem nördlichen Irak führten, trotz AWACS-Luftüberwachung dieses Areals. Seine Analysen werden in dem vorliegenden Text zusammengefasst um anschließend Parallelen für die Bedeutung und Vermeidung von Fehlern in Informationstechnischen Arbeitsabläufen und Projektumsetzungen zu ziehen.

1. Einleitung

Die Lebensdauer von Software verlängert sich immer weiter. Fehler innerhalb einer Software zu beheben, wird mit zunehmendem Alter der Software immer schwerer, zeitaufwändiger und damit teurer. Fehler innerhalb kritischer Systeme, wie beispielsweise Flugcomputer oder medizinischer Geräte können neben finanziellem Schaden erheblichen gesundheitlichen Schaden bei Menschen verursachen. Nicht immer sind die Gründe für Unfälle oder Ausfälle offensichtlich. Selbst wenn die Ursachen für Fehler auf den ersten Blick eindeutig scheinen, lohnt sich doch eine genaue Analyse der Umstände in dem der jeweilige Fehler aufgetreten ist. Um dies hervorzuheben, wird in den folgenden Kapiteln der Abschuss zweier U.S. Army Hubschrauber durch sogenanntes „Friendly Fire“, also durch die eigenen Verbündeten, betrachtet. Bei diesem Abschuss starben alle 26 Insassen der beiden Hubschrauber.

Die Umstände, die zu diesem Abschuss führten, erläutert Scott A. Snook in seinem Buch „Friendly Fire“. Snook selbst hegt großes Interesse an diesem Vorfall, aufgrund einer erhaltenen Verletzung im Zuge seines Militärdienstes durch Friendly Fire. Obwohl der Vorfall, wie sich zeigen wird, großteils nicht durch technisches Versagen oder Softwarefehler ausgelöst wurde, gibt Snooks Analyse doch tiefe Einblicke in die Natur von Fehlern und deren Auftreten bei dem Zusammenspiel verschiedener Akteure. Die Tatsache, dass trotz umfassender und erschöpfender Untersuchungen keinem der Beteiligten die alleinige oder ein Großteil der Schuld an dem Vorfall gegeben werden konnte, gestaltet den Vorfall umso interessanter.

Bei der Analyse der Geschehnisse betrachtet Snook nicht nur die Rollen der unmittelbar Beteiligten, sondern zieht ebenfalls einen weiten Bogen über den Kontext des Vorfalls, sowohl in historisch-politischen Belangen als auch in Hinblick auf strukturelle Faktoren innerhalb des amerikanischen Militärs, die den Vorfall begünstigten. Dieser Text beschränkt sich bei der Beschreibung der verwendeten Militärtechnik sowie der Organisationsstrukturen auf das Minimum um den Vorfall angemessen zu beschreiben. Um die Geschehnisse zur Gänze nachvollziehen zu können, ist es ratsam das Buch zur Lektüre heranzuziehen. Es bietet einen tiefen Einblick in die Entwicklung von Organisationen über einen langen Zeitraum.

Der vorliegende Text betrachtet zunächst in Kürze was passierte. Anschließend wird für die betreffenden Beteiligten betrachtet, warum sie sich so verhielten um zu beschreiben wie der Vorfall passieren konnte. Im Zuge dessen werden für die jeweiligen erkennbaren Fehler Parallelen zu Softwareentwicklung gezogen.

2. Was ist passiert?

Kurz vor Ende des zweiten Golfkrieges, am 7. April 1991, starteten die Vereinigten Staaten „Operation Provide Comfort“ (OPC). Ziel dieser Unternehmung war die Sicherung des Nord-Irak. Dies umfasste den Schutz von Neuansiedlungen kurdischer Flüchtlinge sowie das Durchsetzen der Flugverbotszonen und das Unterdrücken erneuter Aufstände; kurz: die Gewährleistung von Recht und Ordnung. OPC war als „Combined Task Force“ (CTF) konzipiert. Das bedeutet nicht nur verschiedene Einheiten des amerikanischen Militärs, sondern auch verschiedene Einheiten anderer Nationen waren an der Umsetzung beteiligt.

Um in dieser Zusammenstellung zwischen befreundeten und feindlichen Fluggeräten zu unterscheiden, wurde ein „Identification Friend or Foe“ (IFF) System eingesetzt. Dieses besteht aus zwei Komponenten: Einem „Interrogator“ und einem „Transponder“. Jedes Fluggerät trägt einen Transponder. Wenn dieser von einem Interrogator auf der richtigen Frequenz angesprochen wurde, antwortete dieser mit einem numerischen Puls. Es handelt sich hierbei also um ein Challenge-Response-Verfahren. Die Werte der erwarteten Antworten werden über kryptographische Schlüssel festgelegt. Es gibt fünf verschiedene Modi die abgefragt werden können. Modus IV und V sind für die Benutzung von NATO-Streitkräften vorgesehen. Auf die Rolle des IFF im Zusammenhang mit

doi: 10.2313/NET-2014-08-1_07

dem hier beschriebenen Unfall wird in den Abschnitten 3.1 und 3.2 genauer eingegangen.

Am 14. April 1994 gab es drei direkt beteiligte Mitspieler an dem betrachteten Vorfall, auf die sich diese Zusammenfassung konzentriert: Die beiden Black Hawk Hubschrauber, Typ UH-60, der U.S. Army, die Besatzung des E-3B Airborne Warning and Control System (AWACS) und die beiden Piloten der F-15 Eagle Fighter. Sowohl AWACS als auch die F-15 waren Teil der U.S. Air Force. Das AWACS ist eine modifizierte Boeing 707, ausgerüstet mit Systemen zur Überwachung, Kommunikation und vielfältigen Sensoren um diese Aufgaben zu erfüllen. Vereinfachend kann es als fliegender Kontrollturm betrachtet werden. Es verfügt über eine konstant bestehende Leitung zum Bodenkommando.

Am 14. April 1994 startete das AWACS um 0736. Nach Erreichen einer Höhe von 32.000 Fuß, etwa 9.750m, wurden die Systeme gestartet. Dies dauert für gewöhnlich etwa 30 Minuten. Anschließend begab sich AWACS in sein Zielgebiet um den dortigen Luftverkehr zu überwachen. Alle Systeme waren einsatzbereit und es gab keine erkennbaren Probleme.

Die beiden Black Hawks, genannt Eagle 01 und Eagle 02, starteten um 0822. Um 0935 meldeten sie sich bei AWACS an, um in die Flugverbotszone einzutreten. Um 0941 landeten sie in Zakhu um ihre Passagiere aufzunehmen, von wo sie um 0954 wieder starteten und den Start wiederum an AWACS meldeten. Auf dem Rückflug, etwa um 1011, flogen sie in ein tiefes Tal und verschwanden von den Überwachungsmonitoren des AWACS.

Die beiden F-15, genannt Tiger 01 und Tiger 02, erhielten ihr Briefing um 0735 und starteten ihre Maschinen um exakt 0900. Ihre Mission war die Säuberung des Areals von allen feindlichen Flugzeugen und Helikoptern. Nach den routinemäßigen Checks starteten sie um 0935. Sie traten um 1020 in die „Tactical Area of Responsibility“ (TAOR) ein. Zwei Minuten später meldete Tiger 01 Radarkontakt mit einem niedrig und langsam fliegenden Fluggerät 40 Meilen südöstlich seiner Position. Nachdem AWACS keinen Radarkontakt bestätigen konnte, versuchten sowohl Tiger 01 als auch Tiger 02 erfolglos das Fluggerät mit ihrem IFF-Interrogator zu identifizieren. Anschließend starteten sie zur weiteren Untersuchung die Verfolgung. Die nächste Meldung bezüglich Radarkontakts seitens Tiger 01 wurde von AWACS bestätigt. Dem Protokoll folgend führten Tiger 01 und Tiger 02 eine visuelle Zielbestätigung durch. Etwa fünf nautische Meilen vom Ziel entfernt, identifizierte Tiger 01 einen sowjetischen Kampfhubschrauber. AWACS bestätigte die Meldung mit „Copy, Hinds“.

Etwa um 1030 wurden die beiden Hubschrauber, fälschlicherweise als sowjetische Kampfhubschrauber identifiziert, von je einer Rakete getroffen und explodierten. Alle Insassen waren sofort tot. Die Raketen wurden von Tiger 01 und Tiger 02 abgefeuert, nachdem sie eine falsch-negative IFF-Auswertung erhalten hatten und eine fälschliche visuelle Identifikation der Ziele durchgeführt hatten. AWACS, das zuvor mit den beiden UH-60 Helikoptern kommuniziert hatte, hatte nicht interveniert.

3. Wie konnte es passieren?

Im Folgenden wird darauf eingegangen, wie der Abschuss passieren konnte und warum die Schuldigen nicht so einfach zu finden sind, wie es die kurze Zusammenfassung im zweiten Abschnitt dieses Textes vermuten lässt. Hierfür werden für jeden der drei Mitspieler die vorherrschenden Rahmenbedingungen beschrieben und versucht ein Einblick in die jeweils befolgten Routinen zu geben. Anhand jedes Mitspielers und der spezifischen Fehler wird versucht Parallelen zu Fehlerszenarien in der Softwareentwicklung oder beim Umgang mit informationstechnischen Systemen aufzuzeigen. Die Beispiele sind zu diesem Zweck bewusst generell gehalten um die Tragweite nicht anhand von Einzelfällen zu schmälern. Neben den drei bereits genannten Akteuren, wird im Abschnitt 3.4 auf eine Entwicklung innerhalb von großen Organisationen eingegangen, die Snook als „Practical Drift“ bezeichnet.

3.1. UH-60 Black Hawks / Eagle 01 und Eagle 02

Seitens der beiden Helikopter können zwei technische Aspekte genannt werden: Zum einen die technische Fehlfunktion des IFF, wobei diesbezüglich trotz ausführlicher Untersuchungen nicht endgültig geklärt werden konnte, ob der Transponder der Black Hawks oder der Interrogator der F-15 nicht korrekt funktionierte. Zum anderen die technisch mangelhafte Ausstattung in Bezug auf das verwendete Kommunikationssystem.

Das IFF-System wurde von Tiger 01 auf zwei Kanälen abgefragt, nachdem Eagle 01 und Eagle 02 als Radarkontakte erkannt wurden: Zunächst wurde eine Challenge auf Mode I CC gesendet. Mode I war hierbei der generelle Code, mit dem jedes Fluggerät der Koalition antworten sollte innerhalb der TAOR. „CC“ bedeutet hierbei „Correct Code“, also der Code, der für diesen Tag laut Briefing festgesetzt war. Als keine positive Antwort erhalten wurde, änderte Tiger 01 den Modus auf „Auto“. In diesem Modus sendet der IFF-Interrogator kontinuierlich Challenges auf Mode IV. Mode IV war der zweite Modus, auf dem jedes Fluggerät der Koalition antworten sollte. Auch hier erhielt Tiger 01 keine Antwort. Auf zwei verschiedene Challenges erhielt Tiger 01 also jeweils keine Antworten [1]. Diese nicht vorhandenen Antworten können als Falsch-negativ betrachtet werden.

Sowohl AWACS als auch Tiger 01 und Tiger 02 waren ausgestattet mit einem damals unterbrechungssicheren Kommunikationssystem, dem „HAVE QUICK“, auf dessen Gebrauch sie innerhalb der TAOR umschalteten. Es umgeht Versuche die Kommunikation zu stören durch Frequenz-hopping, also das schnelle Wechseln zwischen verschiedenen Frequenzen anhand eines bestimmten Musters, wodurch es gegen Störsignale auf einzelnen Frequenzen nahezu immun ist. Eagle 01 und Eagle 02 waren nicht mit dieser fortschrittlichen Technologie ausgestattet, wodurch sie nicht in die Kommunikation innerhalb der TAOR zwischen Tiger 01, Tiger 02 und AWACS eingebunden waren. Sie verfügten lediglich über Standardsysteme, deren Empfang stark von ungestörten Sichtverhältnissen abhängt. Diese waren nicht gegeben, nachdem Eagle 01 und Eagle 02, den Standardprozeduren folgend, in das Tal eingetreten waren. So verschwanden die

doi: 10.2313/NET-2014-08-1_07

beiden Helikopter nicht nur von dem Radar des AWACS, sondern konnten auch nicht mehr Kommunizieren.

Solche Fehler treten in verteilt entwickelten Systemen oder verteilt arbeitenden Entwicklergruppen auf. Obwohl die Möglichkeit zur sicheren oder effizienten Kommunikation innerhalb des Gesamtsystems gegeben ist, wird sie von einzelnen Teilsystemen aufgrund falscher Anforderungsspezifikation nicht oder nur teilweise genutzt. Wird nun dieser spezielle Fehlerfall von den übrigen Systemen nicht explizit geprüft, geht die Kommunikation innerhalb des Gesamtsystems verloren und das gesamte System gerät in einen instabilen Zustand. Auf Entwicklungsebene passierte genau dieses bei der Entwicklung des „Virtual Case File“, einer Applikation die für das amerikanische FBI zwischen 2000 und 2005 entwickelt werden sollte: Aufgrund schlechter Anforderungsspezifikationen und verteilter Entwicklung standen verschiedene Entwicklerteams teilweise in Konkurrenz zueinander. Es mangelte an klarer Kommunikation zwischen den Unterorganisationen und das Projekt wurde schlussendlich eingestellt [2].

Um einen sicheren und stabilen Betrieb eines Systems zu gewährleisten, ist es also notwendig für jedes Teilsystem geltende Standards zu definieren bezüglich ihrer Schnittstellen mit anderen Systemen. Dabei ist es notwendig nicht nur die Nachbarsysteme zu betrachten, mit denen interagiert wird, sondern ebenfalls einen gelten Status Quo in dem Gesamtsystem zu gewährleisten. Mit einem solchen Vorgehen können auch Ausfälle oder Fehlfunktionen einzelner Untersysteme erkannt und zu einem gewissen Grad kompensiert werden, beispielsweise bei einem byzantinischen Fehler. Ein byzantinischer Fehler beschreibt eine Fehlerklasse in einem System mit mehreren Prozessoren. Spezifisch für diese Fehlerklasse sind unterschiedliche Resultate einer Komponente, abhängig davon auf welchem Prozessor sie arbeitet [3].

Ebenfalls kann so sichergestellt werden, dass bei Erweiterungen oder Änderungen, dieses Subsystem auch von anderen Teilen über einen bestimmten Standard Daten austauschen kann. Hierfür muss für jedes Teilsystem etwaige benötigte Technik zur Verfügung gestellt werden. Ebenso ist es notwendig Kontrollmechanismen zu etablieren, um zu vermeiden, dass verschiedene Systeme auf gleichen Ressourcen arbeiten ohne den Status der anderen Beteiligten zu kennen, falls dieser systemkritisch sein kann.

3.2.F-15 / Tiger 01 und Tiger 02

Bevor auf die begangenen Fehler seitens der beiden F-15 Piloten eingegangen wird, muss deutlich gemacht werden, dass es sich bei diesen beiden Personen um erfahrene Flieger handelte. Beide hatten mehrere tausend Stunden Flugerfahrung, waren ausgeruht und hielten sich ihrem Verständnis nach präzise an die geltenden Prozeduren. Obwohl sie in letzter Instanz den Abschuss auslösten, kann ihnen, wie im Folgenden versucht wird darzulegen, nur eine Teilschuld eingeräumt werden. Die Fehlfunktion des IFF wurde bereits im vorhergehenden Abschnitt näher erläutert, deshalb wird hier darauf verzichtet um Redundanz zu vermeiden.

Aufgrund der negativen IFF-Überprüfung unternahmen Tiger 01 und Tiger 02 eine visuelle Bestätigung des Ziels. Dies ist notwendig, da eine IFF-Challenge nur bestätigen kann, dass es sich bei dem Ziel um ein befreundetes Flugobjekt handelt. Fällt die Abfrage negativ aus, deutet dies jedoch nicht notwendigerweise auf ein feindliches Flugobjekt hin. Tiger 01 sagte im Zuge der Untersuchung des Vorfalls aus, dass nach dem Wechsel der IFF-Einstellung von Mode I CC auf Mode IV kurzzeitig ein verbündetes Flugobjekt an der Position von Eagle 01 und Eagle 02 angezeigt wurde. Diese Anzeige verschwand allerdings kurz nach dem Umschalten wieder. Nach Aussage des Piloten war dies ein häufiger auftretendes Verhalten des IFF in den geflogenen Jets, welches er schon im Verlauf anderer Missionen beobachtet hatte. Deshalb trat keine Verwunderung seinerseits auf [4]. Als Tiger 01 an den Helikoptern vorbeifliegt, ist er etwa 150 Meter über und 300 Meter seitlich von ihnen. Er fliegt in etwa mit einer Geschwindigkeit von 830 km/h. Snook zieht den folgenden Vergleich, um die Größenordnungen und Relationen zu verdeutlichen:

„Imagine trying to tell the difference between a Chrysler Caravan and a Ford Aerostar minivan as one of them passed by you going the other direction […] and each of you is driving 150 miles per hour, and your two lanes were separated by at least three football fields, and the only view you had of the vans was from above and behind, and the vans were painted with appropriate camouflage paint, and you suspect that the passengers in the vans might be armed with the intent of shooting at you after you passed them, and you had a mountain staring you in the face, and you had to make this call while your boss observed your performance from another car driving behind you.“[5]

Dieses Zitat zeigt die wesentlichen Punkte, die zu der fehlerhaften Identifikation führten. Abgesehen von der Distanz und der hohen Geschwindigkeit war der Blickwinkel suboptimal für die Unterscheidung zwischen den beiden Helikoptertypen. Hinzu kommt die Ausstattung der Black Hawks an diesem Tag mit zusätzlichen Treibstofftanks unterhalb der Pontons (der „Flügel“), die die Ähnlichkeit mit dem von Tiger 01 erkannten Modell zusätzlich erhöhten.

Weiterhin war die Vorbereitung der beiden Piloten auf die visuelle Unterscheidung von Hubschraubern nicht mehr aktuell aufgrund zeitlicher Faktoren einhergehend mit den Einsatzplänen der Piloten. Einer der wichtigsten Punkte: Tiger 01 und Tiger 02 wussten nicht, dass verbündete Black Hawks an diesem Tag in der TAOR unterwegs waren. Ohne das Wissen um die Möglichkeit, dass die erkannten Fluggeräte freundlich sein könnten, erwarteten sie feindliche Helikopter. Für F-15 Piloten war es eine inoffizielle Regel, dass bei einer solchen „Säuberungsmission“ eines Areals nur das betreffende AWACS zur Überwachung vor ihnen startet. Aussagen des Piloten von Tiger 01 zufolge sah er tatsächlich den Helikoptertyp, den er angab. Diese optische Täuschung beziehungsweise Fehlinterpretation visueller Daten resultierte aus dem sehr kurzen visuellen Eindruck und seiner unterbewussten Erwartungshaltung, resultierend aus dem Missionsziel und dem unvollständigen Briefing.

doi: 10.2313/NET-2014-08-1_07

Des Weiteren intervenierte AWACS nicht, als Tiger 01 die Radarkontakte meldete. Dies ist ein häufiges Phänomen in Organisationen mit klarer Hierarchie: Solange die Kontrollinstanz nicht interveniert, wird angenommen das Verhalten sei korrekt. So wurden die Piloten von Tiger 01 und Tiger 02 in ihrer Beobachtung es handle sich um feindliche Hubschrauber weiter bestärkt.

Snook spricht noch weitere Faktoren an, wie die anhaltende Rivalität zwischen F-15 und F-16. Den Piloten war bekannt, dass nach ihnen eine F-16 Patrouille das Gebiet sichern sollte. Diesen wollten sie einen sicheren Abschuss nicht überlassen. Kampfpiloten brüsten sich mit der Anzahl ihrer Abschüsse gegenüber anderen Piloten. Im Verlauf des Krieges war die Abschussquote der F-16 aufgrund ihres taktischen Einsatzzweckes höher als die der F-15. Während die F-15 eher für Kämpfe mit anderen Hochgeschwindigkeitsflugzeugen eingesetzt wurde, flogen die F-16 tiefer und hatten als Primärziel langsamere Objekte wie Hubschrauber.

Die Erwartungshaltung der Piloten, gepaart mit unzureichenden Informationen über die vorherrschenden Rahmenbedingungen und einem undeutlichen visuellen Reiz können als Hauptursache für das Verhalten der Piloten angesehen werden. Hinzu kommt das Verlangen den erwähnten F-16 zuvor zu kommen. Abbildung 1 zeigt das Zusammenspiel der einzelnen Teilfaktoren. Das Venn-Diagramm zeigt die visuelle Identifikation als Schnittmenge der drei bereits genannten Teilaspekte. Ebenso beschreibt die Abbildung Stichpunktartig die Gründe für die undeutliche visuelle Erkennung und zeigt ein weiteres Venn-Diagramm, das die Entstehung der Erwartungshaltung darstellt.

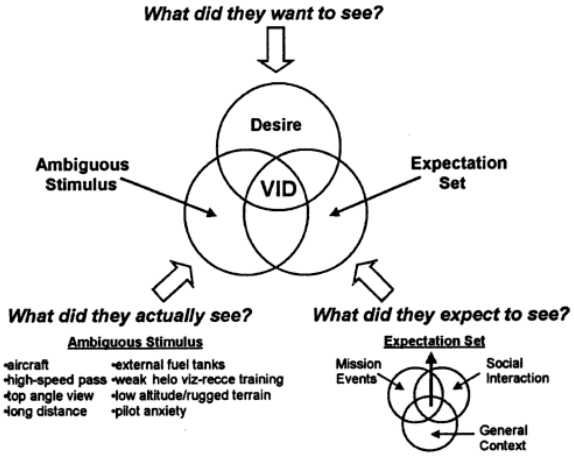


Abbildung 1: Entscheidungsfaktoren der F-15 [6]

Fehler basierend auf falschen Erwartungshaltungen sind menschlich. Da diese nie komplett ausgeschlossen werden können, ist es umso wichtiger ein Umfeld zu schaffen, in dem solche Fehler nicht zusätzlich begünstigt werden. Dazu gehört beispielsweise ausreichend Zeit um mit dem Status Quo vertraut zu werden. So können Verdachtsmomente ausgeräumt werden und fundierte Vermutungen angestellt werden. Obwohl ein Projekt oder das Beheben eines Fehlers innerhalb eines Projekts zeitkritisch ist, kann es auch in der Softwareentwicklung fatale Auswirkungen haben, eine Fehlerursache aufgrund von Vermutungen vermeintlich zu beheben ohne genau zu analysieren, ob der Fehler wirklich an dieser Stelle auftritt. Neben unnötigen Mehraufwand können so weitere Fehler in den Code eingebracht werden aufgrund unsauberen Arbeitens oder entstehender Redundanz.

Die falsche visuelle Identifikation der Ziele zeigt weiterhin die Notwendigkeit ausreichend zuverlässiger Kompensationsprozeduren bei Ausfall einer Schlüsselfunktion. Die Fehlerrate unter den gegebenen Umständen ist bei rein visueller Identifikation erwartungsgemäß viel höher als die Wahrscheinlichkeit eines Falsch-positiven oder –negativen Ergebnisses bei Abfrage mit Hilfe eines IFF-Systems. Um dies zu umgehen, kann auf den Einsatz von „fail-safe“-Systemen zurückgegriffen werden. Diese verhindern Ausfälle kritischer Komponenten zwar nicht, gewährleisten jedoch, dass im Falle eines Fehlers das System nicht weniger sicher ist, als wenn dieser Fehler nicht auftritt [7]. Ebenso zeigt die Gleichgültigkeit, mit der Tiger 01 die Fehlfunktion bei Umschaltung auf Mode IV hinnimmt, die Wichtigkeit von nachhaltiger Wartung solch kritischer Systeme.

3.3.AWACS

Während festgestellt wurde, dass Eagle 01 und Eagle 02 mit veraltetem Equipment flogen und in dem verwendeten IFF ein Fehlverhalten auftrat, muss bei der Rolle der Crew des AWACS nachgeforscht werden, warum diese nicht intervenierten, als Tiger 01 zwei unbekannte Radarkontakte meldete.

Betrachtet man die Mitglieder der AWACS-Besatzung einzeln, ist auch hier festzustellen, dass jeder von ihnen für seine Stellung hoch qualifiziert war. Jedes Mitglied verfügte über ausreichend Erfahrung auf seinem spezifischen Gebiet, war ausgeruht und voll einsatzfähig. Liegt das Augenmerk allerdings auf der Besatzung als Ganze, ist festzustellen, dass sie in dieser Zusammensetzung keinerlei Erfahrung aufweisen konnte. Jede AWACS-Besatzung muss vor einem Einsatz mindestens zwei komplette Simulationsläufe erfolgreich durchführen. Dies war jedoch nicht geschehen aufgrund verschiedener Faktoren wie Krankheit, verspätetes Eintreffen in der Operationszone, Rotationspläne der Einsatzziele und Ähnlichem. Snook unterscheidet zwischen fünf verschiedenen Graden der Performanz eines Teams [8]:

- „Working Group“: Mitglieder agieren primär um Informationen zu teilen. Davon abgesehen gibt es keine tiefer greifende Zusammenarbeit
- „Pseudo-Team“: Mitglieder versuchen zusammen zu arbeiten, dabei sinkt jedoch die Leistung jedes einzelnen Mitgliedes, sodass die Gesamtleistung niedriger als die Summe der Einzelleistungen ausfällt
- „Potential Team“: Mitglieder arbeiten teamorientiert, aber nicht optimal; die Performanz ist etwa die einer Working Group
- „Real Team“: Eingespieltes Team, Mitglieder arbeiten auf ein gemeinsames Ziel hin und fühlen sich für die Leistung der Gruppe verantwortlich
- „High Performance Team“: Erfüllt alle Kriterien des „Real Team“, Mitglieder sind darüber hinaus

doi: 10.2313/NET-2014-08-1_07

an der Weiterentwicklung der übrigen Gruppenangehörigen sehr interessiert

Aufgrund des fehlenden gemeinsamen Trainings und daraus resultierender nicht existenter Erfahrung in dieser spezifischen Zusammensetzung, ist die AWACS-Besatzung an diesem Tag als „Pseudo-Team“ einzustufen. Obwohl also jedes Mitglied selbst qualifiziert genug war für die Mission, konnte die Gruppe als solche sie nicht optimal erfüllen.

Hinzu kommen Gewohnheitsfaktoren der Mitglieder: So gibt ein Mitglied bei einer Befragung zu Protokoll, er habe immer ein Buch dabei, da er nirgendwo soviel lese wie während eines Fluges. Ebenso gab es keine offiziellen Protokolle für die Behandlung von U.S. Army Flugobjekten innerhalb der TAOR seitens der U.S. Air Force AWACS. Ein Mitglied der AWACS-Besatzung gibt an, dass Angehörige der U.S. Army sich manchmal anmeldeten, wenn sie in die TAOR eintraten, es manchmal aber auch unterließen. Daraus resultierend gab es keine feste Vorgehensweise bezüglich der Behandlung der Hubschrauber innerhalb der TAOR. Somit fühlte sich keines der Mitglieder der Besatzung dafür zuständig, Tiger 01 und Tiger 02 über die Anwesenheit der Hubschrauber aufzuklären.

Dies wird besonders dadurch deutlich, dass nach Abbruch des Radarkontakts zwischen Eagle und AWACS bei Eintritt in die Berge zwar ein Marker seitens der Besatzung gesetzt wurde, der die letzte Position auf dem Radar anzeigt, dieser aber nach 60 Sekunden verschwand weil sich niemand dessen annahm. Ähnlich wie Tiger 01 und Tiger 02 verließ sich unterbewusst jedes der Mitglieder darauf, dass sein jeweiliges Kontrollorgan auf Fehler aufmerksam machen würde.

Es kommt weiterhin erschwerend dazu, dass die Stellung der Besatzung innerhalb der Air-Force nicht die beste ist. Obwohl die Befehlshierarchie darlegt, dass die Besatzung als Kontrollorgan theoretisch einen signifikanten Einfluss auf die Aktionen von Kampfpiloten hat, zeigt die Realität, wie Snook beschreibt, ein anderes Bild. Entgegen der theoretischen Einbindung in die Befehlsstrukturen, betrachten Kampfpiloten jede andere Klasse von Angehörigen der Air-Force als minderwertig oder Piloten zweiter Klasse. So gibt ein Angehöriger der AWACS-Besatzung zu Protokoll, dass er für gewöhnlich in die Entscheidungen, die ein Kampfpilot trifft, nicht eingreift, da diese Einmischung ohnehin meist ignoriert wird [9]. In Anbetracht dieser gesellschaftlichen Strukturen ist es nicht weiter verwunderlich, unter Mitberücksichtigung der bereits genannten Gründe, dass niemand innerhalb der AWACS-Besatzung intervenierte oder sich dazu berufen oder verpflichtet fühlte, in irgendeiner Art und Weise der visuellen Bestätigung durch die Kampfpiloten Tiger 01 und Tiger 02 zu widersprechen.

Es bleibt also festzuhalten, dass die AWACS-Besatzung zwar versäumte zu intervenieren und dadurch ein tragischer Unfall begünstigt wurde. Dennoch muss beachtet werden, dass aufgrund fehlender Ablaufprotokolle seitens der Einbindung von U.S. Army Operationen in U.S. Air Force Operationen kein Mitglied der Besatzung voll dafür verantwortlich gemacht werden kann. Weiterhin wurde durch die gesellschaftliche Struktur innerhalb der Air-Force die nicht-Einmischung seitens der Besatzung gerechtfertigt.

Solche Fehler sind typisch bei großen Softwareprojekten. Es ist gängige Praxis knappe Zeitpläne durch erhöhte Entwicklerzahlen zu kompensieren. An dieser Stelle schlägt jedoch direkt die Auswirkung des „Pseudo-Teams“ zu: Anstatt die Performanz der Gruppe zu erhöhen, werden bereits etablierte Entwickler in ihrer Arbeit verlangsamt durch Einarbeitung der neuen Kollegen. Nicht selten werden Entwickler hierfür von ihrer eigentlichen Aufgabe abgezogen und als Leiter kleinerer Sub-Teams eingesetzt. Mit dieser neuen Rolle sind sie nicht vertraut und begehen Fehler, die vermeidbar wären. Die neuen Kollegen wiederum brauchen Zeit um sich in die Unternehmensstrukturen, die Programmierrichtlinien und das neue soziale Umfeld einzufinden. Die jeweiligen Zuständigkeiten sind nicht mehr klar gegliedert und undurchsichtiger, je größer das jeweilige Softwareprojekt ausfällt. Dies erhöht signifikant die Fehlerquoten durch Einbindungskonflikte, Programmierstile und mangelnde Kenntnis der Projektumgebung, sowie mangelnde Identifikation mit den Zuständigkeiten und Verantwortlichkeiten der individuell eingenommenen Rolle [10].

Dasselbe Phänomen ist nicht nur bei dem Einsatz externer Entwickler, sondern ebenfalls bei ad hoc rekrutierten Testern zu beobachten. Sie sind nicht vertraut mit Testprotokollen, legen unterschiedlichen Wert auf exploratives Testen und sind andere Aktionsabläufe beim Finden von Fehlern gewohnt. Im schlimmsten Falle sind die Tester selbst unerfahren, da übersehen wird, dass effektives und effizientes Testen Fähigkeiten sind, die es zuerst zu erlernen gilt, bevor sie umgesetzt werden können [11]. Dies zeigen auch die Ergebnisse diverser Studien bezüglich der Erfolgsfaktoren im Projektmanagement und somit auch im Management von IT-Projekten. Die drei wichtigsten Faktoren hierbei sind Kommunikation, Zieldefinition und Qualifikation der Mitarbeiter [10].

3.4.Practical Drift

Um die dynamische Natur der Zusammenhänge zwischen Unterorganisationen in einem größeren Kontext zu erklären, führt Snook eine Theorie ein, die er als „Practical Drift“ bezeichnet. Kurz gefasst besagt er in dieser Theorie, dass Angehörige einer Organisation immer einen billigeren, einfacheren oder effizienteren Weg finden werden Aufgaben zu erledigen und deshalb von vorgegebenen Richtlinien oder Prozessen abweichen werden. Im Zuge dieses Vorganges stabilisiert sich ein momentan instabiles System aus Aktionsmotiven und Organisationsbindungen selbst. Er bezeichnet dieses Verhalten kurz als „[…] the slow steady uncoupling of practice from written procedure“ [12]. Dies führt dazu, dass Vorgehensweisen, die sich lokal als effektiv und effizient erweisen, durch Wiederholung legitimiert werden. Es ist eine direkte Folge aus dem Umstand, dass aufgestellte Regeln nicht immer direkt auf die vorherrschende Situation anwendbar sind oder in dieser spezifischen Situation suboptimal erscheinen. Dies liegt zumeist an einer Überdefinition der Regeln.

Kommt nun zu diesem Phänomen des „Practical Drift“ eine plötzliche enge Bindung zwischen verschiedenen Unterorganisationen, die im Laufe der Zeit ihre eigenen

doi: 10.2313/NET-2014-08-1_07

Versionen der Prozesse und Richtlinien durch Wiederholung etabliert haben, entsteht ein gefährliches Moment, welches Snook als weitere Ursache für den Abschuss der beiden UH-60 Black Hawk Helikopter angibt. Dies ist einer der besagten instabilen Momente innerhalb eines Systems.

Hätten alle Beteiligten die gleiche Auffassung der Prozesse gehabt, wäre der Vorfall wahrscheinlich nicht eingetreten. Hätten sich die Helikopter der U.S. Army bei jedem Flug bei AWACS an- und abgemeldet, wäre schnell aufgefallen, dass es seitens der Army nicht klar war, dass innerhalb der TAOR geänderte IFF-Codes genutzt wurden und dieser Umstand hätte behoben werden können. Ebenso wären die möglichen Auswirkungen des teilweisen Einsatzes veralteter Technik innerhalb der Black Hawk Helikopter wahrscheinlich eingehender betrachtet worden. Wäre die Army korrekt und umfassend in die Verlaufspläne der Air Force eingebunden gewesen, hätten Tiger 01 und Tiger 02 innerhalb ihres Briefings die Information erhalten, dass an diesem Tag vor ihrem eigenen Start bereits zwei Helikopter gestartet waren.

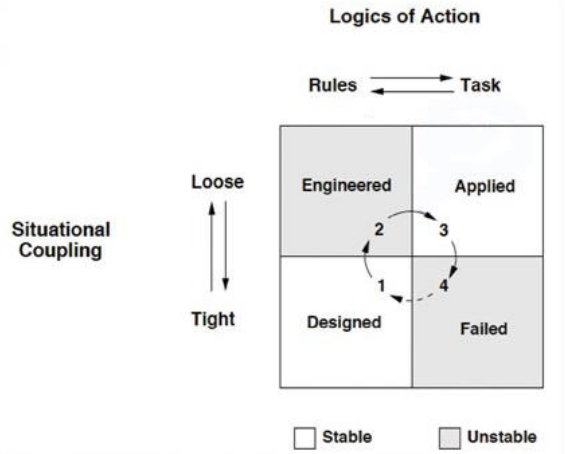


Abbildung 2: Practical Drift [13]

Abbildung 2 zeigt Snooks Modell in drei Dimensionen: Aktionslogik, situationelle Verbundenheit und Zeit. Aktionslogik kann entweder auf Regeln oder die zu bewältigende Aufgabe ausgelegt sein. Die situationelle Verbundenheit ist entweder eng oder lose. Die Zeit wird anhand der Zahlen 1 – 4 dargestellt. Quadranten 1 und 3 kennzeichnen hier stabile Zustände, die Aktionslogik passt zur vorherrschenden Situation. In den Quadranten 2 und 4 ist das System instabil, lose Bindungen sind nicht vereinbar mit regelbasierten Aktionen, ebenso wenig wie enge Bindungen mit aufgabengetriebenen Aktionen.

Was das bedeutet, erklärt folgendes Beispiel: Um den Prozess der Softwareentwicklung zu optimieren, soll ein neuer Deployment-Prozess entwickelt werden. Um nicht für vage Formulierungen zur Verantwortung gezogen werden zu können, definiert der Ersteller dieses Prozesses alles so präzise wie möglich und orientiert sich bei der Erstellung an diversen Worst-Case-Szenarien. Dadurch wird der Prozess für die praktische Umsetzung übertrieben präzise definiert. Dies ist der erste Quadrant, die Regeln sind streng definiert, die Teilprozesse eng gekoppelt.

Als der Prozess fertiggestellt ist und in den ersten Projekten eingesetzt wird, geht er in den zweiten Quadranten über. Das Vorgehen ist nach wie vor regelorientiert, aber die einzelnen Prozessgruppen sind nicht mehr so eng gekoppelt wie in der Theorie gedacht. Dieser Zustand ist instabil und geht daher langsam in den dritten Quadranten über: Im Laufe der Zeit werden für die Anwender hinderliche Teile von ihnen weggelassen oder für ihre Zwecke optimiert. Dieser Zustand ist wiederum stabil und kann über Jahre andauern. Wie Snook beschreibt, werden die angepassten Teilprozesse innerhalb der Unterorganisationen die neuen Standards.

Diese Situation bleibt jedoch nur solange stabil, bis etwas passiert, das Unterorganisationen dazu zwingt, eng miteinander zusammen zu arbeiten. In dem genannten Beispiel könnte es sich hierbei etwa um einen Fehler bei der Einbindung automatisierter Tests in den Deployment-Prozess handeln. Müssen nun zwei Unterorganisationen miteinander zeitkritisch arbeiten, während jede im Laufe der Jahre ihre eigenen Prozesse etabliert hat, ist es wahrscheinlich, dass irgendein größerer Fehler auftritt. Das System befindet sich nun im vierten Quadranten. Um das Wiedereintreten solcher Fehler zu vermeiden, werden nun neue Prozesse aufgesetzt, das System stabilisiert sich wieder in Quadrant eins. Wurden aus der Vergangenheit keine Lehren bezüglich der Etablierung von Richtlinien gezogen, beginnt der Kreislauf nun wieder von Neuem mit den neu eingeführten Regeln.

Dieses Beispiel, basierend auf Snooks Theorie, lässt folgende Schlussfolgerungen zu:

- Was Snook als „Practical Drift“ bezeichnet, ist unvermeidlich, der Mensch versucht seine Arbeit zu optimieren
- Mehr oder umfangreichere Regeln und Prozesse begünstigen dieses Verhalten
- Im Idealfall sollte der einfachste Weg etwas zu tun der richtige Weg sein
- Prozesse müssen zusammen mit denjenigen entwickelt werden, die sie später befolgen sollen
- Prozesse und Richtlinien müssen sich im Laufe der Zeit mit der gängigen Praxis ändern

Dieses Verhalten kann in jeder Organisation beobachtet werden. Der IT-Sektor bildet hier keine Ausnahme. Firmeneigene Coding-Standards werden ignoriert um schnell einen Fehler zu beheben. Lokale Administratorrechte werden an nicht ausreichend qualifizierte Nutzer vergeben, um Zeit bei der Installation von Programmen zu sparen. Firmenlaptops werden mit nach Hause genommen um weiter zu arbeiten und gehen verloren oder werden gestohlen. So wurde beispielsweise Ende 2012 einem Mitarbeiter der NASA sein Laptop gestohlen. Der Rechner selbst war zwar mit einem Passwort geschützt, die Festplatte selbst war jedoch unverschlüsselt [14]. Im Grunde genommen kann für jede geltende Regel an dieser Stelle ein Verstoß gegen selbige genannt werden. Wie bereits von Snook genannt ist der Antrieb zur Dehnung oder direkten Missachtung dieser Regeln Bequemlichkeit, Zeitersparnis oder ein Kostenfaktor. Und durch lange Zeit ausbleibenden Schaden werden diese neuen Auslegungen die akzeptierten Regeln.

doi: 10.2313/NET-2014-08-1_07

4. Zusammenfassung

Abschließend ist erneut festzuhalten, dass Fehler passieren. Die hier betrachteten Fehlerklassen reichen von technischen Defekten und mangelhafter technischer Ausstattung über menschliches Versagen aufgrund unzureichender Information und daraus resultierender Erwartungshaltung bis hin zu Passivität und suboptimaler Performanz aufgrund von unerfahrenen Teams. Keine dieser Fehlerklassen kann innerhalb von komplexen Systemen, bei deren Betrieb Menschen beteiligt sind, komplett ausgeschlossen werden.

Unter dieser Prämisse ist es sinnvoll die Rahmenbedingungen anzupassen um die Wahrscheinlichkeit des Auftretens dieser Fehler zu minimieren. Wie der „Practical Drift"-Ansatz zeigt, sind starre Strukturen hierfür kontraproduktiv, da sie auf lange Sicht betrachtet lediglich das Abweichen oder Umgehen dieser Strukturen fördern. Prozesse und Richtlinien müssen analog zur Entwicklung des Umfeldes weiterentwickelt werden, um auf geänderte Umstände weiterhin anwendbar zu sein. Korrekte und ausführliche Kommunikation zwischen Unterorganisationen, handle es sich hierbei um unterschiedliche Abteilungen innerhalb eines Unternehmens oder um Unterklassen eines Softwaresystems, sind notwendig um Störungen frühzeitig zu entdecken und darauf reagieren zu können.

Das Verhalten der AWACS-Besatzung zeigt, dass streng gegliederte Hierarchien innerhalb von Organisationen ebenso Fehler begünstigen können. Durch das Verlassen auf die jeweils nächste Kontrollinstanz ohne Intervention werden Fehler hierbei zwar von allen Beteiligten wahrgenommen, jedoch von niemandem darauf reagiert, bis es schließlich zu spät ist. In System mit strengen Hierarchien ist es daher notwendig für die einzelnen Ebenen genaue Verantwortlichkeiten zu definieren, sodass zu jedem Zeitpunkt jedem Beteiligten bewusst ist, wer sich worum kümmern muss.

5. Quellen

[1] Scott A. Snook. *Friendly Fire – The accidental shootdown of U.S. Black Hawks over northern Iraq*. Fourth Printing, and first paperback printing, 2002, Princeton Paperbacks, ISBN 0-691-09518-3, S. 60

[2] Glenn A. Fine. Congressional Testimony. http://www.justice.gov/oig/testimony/0502/final.pdf, aufgerufen am 19.3.2014

[3] L. Lamport, R. Shostak, und M. Pease: *The Byzantine Generals Problem.* In: ACM Trans. Programming Languages and Systems. 4, Nr. 3, Juli 1982, S. 382-401

[4] Scott A. Snook. *Friendly Fire – The accidental shootdown of U.S. Black Hawks over northern Iraq*. Fourth Printing, and first paperback printing, 2002, Princeton Paperbacks, ISBN 0-691-09518-3, S. 60-61

[5] Scott A. Snook. *Friendly Fire – The accidental shootdown of U.S. Black Hawks over northern Iraq*. Fourth Printing, and first paperback printing, 2002, Princeton Paperbacks, ISBN 0-691-09518-3, S. 88

[6] Scott A. Snook. *Friendly Fire – The accidental shootdown of U.S. Black Hawks over northern Iraq*. Fourth Printing, and first paperback printing, 2002, Princeton Paperbacks, ISBN 0-691-09518-3, S. 97

[7] David B. Rutherford Jr., *„What do you mean – it's fail safe? Evaluating Fail-Safety in processor-based vital control systems"*, 1990 Rapid Transit Conference, http://www.billpetit.com/Papers/Petit017.pdf, aufgerufen am 23.3.2014

[8] Scott A. Snook. *Friendly Fire – The accidental shootdown of U.S. Black Hawks over northern Iraq*. Fourth Printing, and first paperback printing, 2002, Princeton Paperbacks, ISBN 0-691-09518-3, S. 106

[9] Scott A. Snook. *Friendly Fire – The accidental shootdown of U.S. Black Hawks over northern Iraq*. Fourth Printing, and first paperback printing, 2002, Princeton Paperbacks, ISBN 0-691-09518-3, S. 132

[10] C. Engel, A. Tamdjidi, N. Quadejacob. *Ergebnisse der Projektmanagement Studie 2008- Erfolg und Scheitern im Projektmanagement.* Studienergebnisse, http://www.gpm-ipma.de/fileadmin/user_upload/Know-How/Ergebnisse_Erfolg_und_Scheitern-Studie_2008.pdf, S. 8

[11] J. Vargas, J. Córdoba. *10 Best practices for an effective testing & QA implementation.* http://www.softtek.com/newsletter/item?id=16&item=77, aufgerufen am 19.3.2014

[12] Scott A. Snook. *Friendly Fire – The accidental shootdown of U.S. Black Hawks over northern Iraq*. Fourth Printing, and first paperback printing, 2002, Princeton Paperbacks, ISBN 0-691-09518-3, S. 194

[13] Scott A. Snook. *Friendly Fire – The accidental shootdown of U.S. Black Hawks over northern Iraq*. Fourth Printing, and first paperback printing, 2002, Princeton Paperbacks, ISBN 0-691-09518-3, S. 186

[14] U. Ries. *Datenleck durch Laptop-Klau bei der NASA.* http://www.heise.de/security/meldung/Datenleck-durch-Laptop-Klau-bei-der-NASA-1750677.html, aufgerufen am 19.3.2014

What does your mobile phone leak - Unique in the crowd and predictable

Matti Lorenzen
Betreuer: Heiko Niedermayer
Seminar Future Internet SS2014
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: matti.lorenzen@tum.de

ABSTRACT

In this paper, the fact that our mobile phones tell more about us than most people think will be presented. Not only the integrated hard drive but also the archive of the provider or other parties involved with the usage of the cell phone are filled with valuable personal data. This data is highly sensitive and is already being used for personalized marketing and other purposes. After discussing which kind of data is gathered by the cell phone, focus will be laid on the accessability of this data which is often enough to identify the user uniquely in a crowd and even to predict his personality. However, this is not only to be seen as a challenge but also as an opportunity for a better interaction between human beings and machines in the future.

Keywords

Big data - Anonymity - Privacy - Uniqueness - Adaption - Predictability - Accessability - Personalization

1. INTRODUCTION

With almost seven billion global subscriptions [10], cellular services are available to more than 90% of the earth's population. With coverage around the globe, it seems impossible to determine the exact ways in which users are interacting, especially if only the raw data is reviewed. However, studies show, that data leaked by the cell phone is indeed so distinctive that the user is becoming unique in the crowd. In fact, this may be taken even further: Many phone users spend more time with their phones than with any other person; the way we use our cell phones tells a lot about us and makes it possible to predict our personalities with precision of up to 80%. If we are that predictable, will our cell phones adapt to our feelings, emotions or personal traits in the future? What consequences lie ahead if we really are that predictable and do not take precautions in order to protect our personal data? Are we actually in danger? These questions will be answered in the course of this paper.

First, the data gathered by the phone will be presented and it will be discussed who is able to access this information. After presenting the data gathered in a self-study, focus will be laid upon the phone user's personality; how it can be determined and how that information can be used. In order to conclude, the question, whether individuals are appearing uniquely in a crowd, will be addressed and possible solution to reclaiming privacy will be introduced.

2. WHAT DOES MY MOBILE PHONE LEAK?

In the following chapter, the data collected by the cell phone will be described. Furthermore, the question whom is able to access the data obtained will be answered. The amount of people that have access to this information is stunning, however, we often allow our cell phones to leak more than necessary.

2.1 What kind of data does my cell phone gather?

The data collected by the cell phone is immense[5]. In this paper, focus will be laid especially on spatio-temporal data, so location information will be discussed thoroughly. Additionally, other collected information will be discussed although it will not be relevant for the chapters regarding predictability and uniqueness as they already contain personal identifiers.

2.1.1 Location information

The cell phone's location and so, in most cases, the user's location can be detected in different ways. The probably best known possibility is tracking through GPS which is highly accurate with an error of less than five meters.

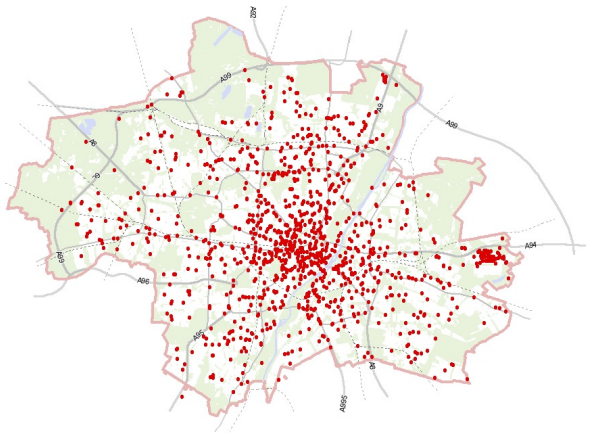


Figure 1: GSM base stations in Munich[23]

However, on most modern cell phones, the GPS signal can be deactivated by the user misleading him to the assumption that his location is untraceable. However, if the cell phone is connected to a wireless network, the location can be resolved

doi: 10.2313/NET-2014-08-1_08

out of the phone's IP address in this network which leads to an accuracy of at least 100 meters.

While the cell phone's wireless connection receiver may also be deactivated, the GSM signal cannot be deactivated without the loss of key functionalities. Normally, at least three GSM base stations are within the cell phones signal range, in cities the amount is increased by a big margin as visualized in Figure 1. There, the GSM base stations of Munich are depicted. The strength of the signal received by the cell phone from the different stations varies and so, the user's location is calculable accordingly. This happens during the stand-by mode. When calls are made, the cell phone ID is recorded at the station closest by which leads to a higher accuracy. Tracking through different types of wireless networks such as UMTS or LTE is working in a similar way.

2.1.2 Phone information

In Europe, more than 50% of the cell phones used are smartphones. The mostly used operating systems, iOS and Android (In this paper, focus will be laid upon the operating systems provided by Apple and Google as they hold a market share of over 90%.), require the user to sign up for an account at Apple or Google, respectively, where he has to specify information including his name, date of birth, email address, credit card information and phone number. Within these accounts, contacts and the calendar are administrated in order to be synchronized with other devices. Hence, the contact information of friends and family are also saved. If the user decides to connect his smartphone with a social network, increased data is added due to the synchronization of the address book or the calendar. Additionally, every telephone has a unique serial number, the so called IMEI which is used in order to identify the cell phone.

2.1.3 Camera and microphone access

The camera and microphone integrated in the cell phone may not only be used by the cell phone's owner. While requirements to access the camera and the microphone of programs like Skype on the phone are understandable, many other applications, the user would not normally have thought of, may access any of these features as well. The images taken and the audio recorded may be saved on the cell phone's internal hard drive or uploaded to online cloud services. Additionally, pictures taken or sound recorded may gather a lot of metadata such as date recorded, location recorded, camera preferences which will automatically be shared with others when we share our pictures.

2.2 Who can access these information?

The data gathered on the mobile phone can be accessed by the cell phone provider, the operating system, the applications installed on the device and other parties that may access this information without permission. The provider monitors the telephone's usage and has access to all the data collected [14] while using the UMTS or GSM network, most importantly the location respective to the time, calls and texts received and made or send, respectively. However, with the personal information the provider already obtained by signing the individual to a phone contract and the data gained by the phone usage, the provider has a deep insight into that client's profile. Profiles may include over 800 attributes including ethnicity, age, income level and frequently visited places. According to media releases [11], aggregated user profiles are likely to be sold by the provider to external enterprises due to the high value of this information regarding marketing aspects. [17] [20]

As the operating system is vital to the phone functioning correctly, complete access to all data gathered on the cell phone is required. Additionally, the operating system's manufacturer reserved its right to access the call and messaging protocol. Hence, Google [15] or Apple, respectively, know with whom their client interacted, when he interacted and how frequently he interacts. This information may be sold as well, as described before. In fact, according to a study held in 2013 [9], each unique user has an average value of about $24 to Google. Those who are not only using the search engine but also own a smartphone operated on Android provide even more data to the company and so, their value is increased.

Upon installation of a new application to his smartphone, the user is usually asked to give permissions to this certain application. The requested permissions often go beyond the features required in order to work correctly. As an example, the Deutsche Bahn's mobile app for Android 4.4 requires the user to reveal his call and messaging history. Although permissions may be revoked, the user should pay attention to the permissions granted and weigh them up to the functionality obtained.

In summer 2013, the documents revealed by former NSA[1] employee Edward Snowden caused a global surveillance scandal. According to his information, the NSA was able to gain information of almost every phone user they intended. Even politicians' phones where among the ones spied upon. Additionally to the call and messaging protocol, also their contents were accessible and the phone's microphone and camera were activated remotely [13] in order to gather intelligence. Only weeks later, it was revealed that the British GCHQ[2] was able to do exactly the same. It is not yet known how much data has been obtained by spying or who has also been able to attack the cell phone despite of all privacy laws.

2.3 Personal data example

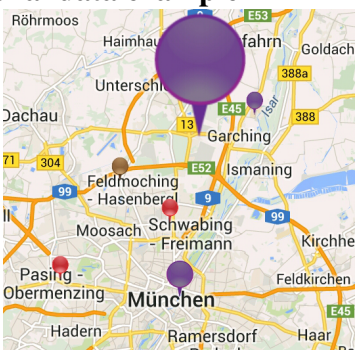


Figure 2: Locations recorded via Track My Life[24]

During the past month, I wanted to find out whether the data gathered by my cellphone is indeed precise enough to

[1] American National Security Agency
[2] British equivalent to the NSA

doi: 10.2313/NET-2014-08-1_08

identify me as the user corresponding to the phone's dataset.

By using the application "Track my Life" [24] created by a student of the Technische Universität München, my location was recorded whenever I spent more than two hours at the same place. The locations most frequently tracked were my apartment, the university buildings, the gym and friends' places. In Figure 2, these top locations are visualized. The size of the points is proportional to the amount of data gathered at that location.

While over 10.000 people are studying at the university campus I am going to, only few of them live in an area of 500 meters to my home. But still, the amount of possible persons corresponding with this dataset is at about 10 people.

By adding more locations into the perspective, like the friends visited or the international group I am attending on a weekly basis, I am soon the only possible match for the mobility trace given.

This already shows that the data needed in order to define mobility traces is not as big as one might initially expect. However, in order to link a person correctly, knowledge of the person is required. This knowledge may be asymmetric as for many persons information is accessible through social networks. This topic will be discussed more thoroughly in chapter five.

Further, I reviewed my call and messaging protocol during the past month. This revealed, that interaction is made on average every three hours[3] without consideration of sleep. The contacts with whom I interacted are spread all across the country, some of them were even outside of Germany while the interaction took place. Factors as the time between interactions and the entropy of contacts can be used in order to predict traits of my personality as will be described more thoroughly in chapter three.

3. PERSONALITY PREDICTION

No matter whether a person is waiting for a train, enjoying his or her lunch break or at work - the mobile phone is always close by. In fact, many people interact more with their cell phones than with other people.

When other people spend a lot of time with us, they can usually identify our personality quite well by the way we are interacting with them. Does our interaction with our mobile phones provide enough data that our personality traits can be predicted only by review of the data gathered?

Smartphones are used for various purposes; among the most frequently used services are messaging, taking photos, accessing social networks and playing games. By analyzing the content of these messages, reviewing their activities in social networks or evaluating the frequency of games played, it would not be too hard to predict the personality quite precisely. In Figure 3, the most frequent usages as determined in a study held in the USA in 2012 are visualized. Especially the evaluation of sites liked on Facebook are accessable to

[3]Calculated based on 227 interactions within the time between 03/22/14 and 02/22/14

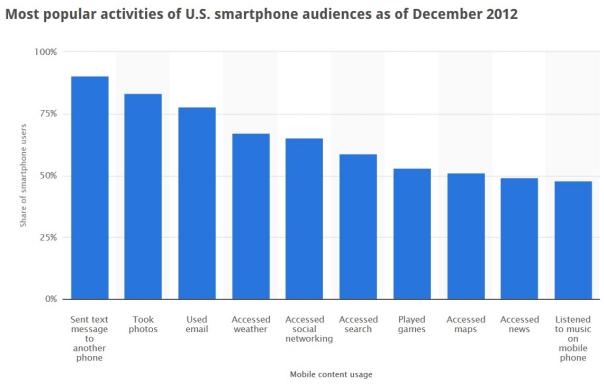


Figure 3: Mobile content usage[22]

everybody and a very strong indicator in order to predict personality traits. While gender and race were identified in over 90% of the cases, sensitive information like religion, sexual- or political orientation were predicted correctly in most cases as well. However, only the provider or the operator are allowed to gather these information legally and privacy concerns would arise most certainly.

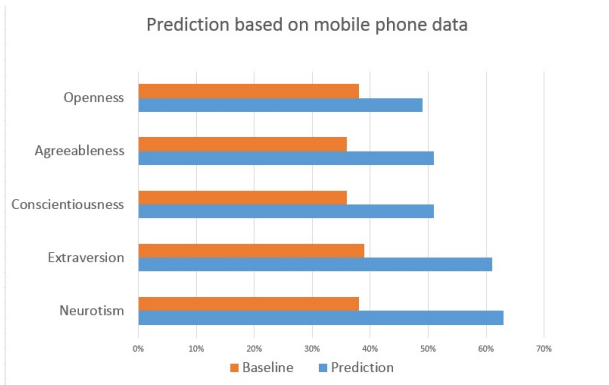


Figure 4: Big five personality traits and the comparision of prediction rates

However, researches have come up with a different approach to predict personality by reviewing information that is easily accessible and less private[19, 2].

Psychologists define five major personality traits [6], also known as the "Big Five." They are openness, agreableness, conscientousness, extraversion and neurotism. During a study at the Massachusetts Institute of Technology [2], a prediction upon persons' personalities was made only by reviewing data gathered by cell phones. While the conclusion that a person that is frequently interacting with his mobile phone by writing messages to friends or calling them is more likely to be more open than others appears to be obvious, other connections between cell phone usage and psychology are far less trivial.

In this study, a distinction between interacting by calling somebody or by texting somebody was made, as an open, extroverted individual is more likely to use a direct form of communication than one who is not. These interactions have

doi: 10.2313/NET-2014-08-1_08

been analyzed scientifically by reviewing data regarding the regularity and the diversity of phone usage, the individual's spatio behaviour, active behaviour and basic phone use. As useful indicators, average and variance of inter-event time for calls and/or messages, the entropy of contacts or places and the percentage of initiated texts or calls was defined.

Those indicators have been linked to the personality traits initially presented, giving the researchers the possibility to predict the person's personality. The predictions after review of this data have proven to be on average 42% better than random. This is visualized in Figure 4. When taking into account information like gender or age, the accuracy was improved even further. In order to make these predictions, only basic mobile phone data was reviewed. Hence, predictions can not only be made for persons owning a smartphone but also those who stick to more classical models.

These predictions, however, can be used for personalized marketing as the marketer can target the right kind of customer with the right kind of advertisement in order to trigger a purchase. A person that is highly agreeable is more likely to be effected by a marketing campaign than one who is not. Combined with the other information gathered by the cell phone like location, mobility trace and check-ins in social networks, it offers a great opportunity to advertise their store which is located right on the costumer's usual way in order to get to his work place or to advertise places similar to the ones visited such as ski resorts or amusement parks.

Further, not all persons interact digitally the same way as they would do in "real life". On the one hand, a shy person tends to interact via texting instead of talking to the person directly. On the other hand, a socially active person may not interact a lot via cell phone as he or she does not have the time or the desire to do so. The criteria on which the personality traits are mapped surely correspond with many personality traits of many people but most definitely not with all of them.

Still, the prediction has proven to be accurate in more than two thirds of all cases as a study held in Barcelona [18] revealed in which the data of 600 volunteers was collected during six months. The personality traits predicted were compared to the results of a personality test every participant took in the beginning of the experiment. Future research will have to improve these numbers in order for personality prediction to be seen as a key factor. In order to do so, the criteria reviewed will have to be extended or other sources of information will have to be accessed.

4. PERSONALITY ADAPTION

As seen before, every person interacts differently with his mobile phone. When people interact with each other, they are adapting to other people's roles in society as they will talk differently to a police officer, a famous actor or the nice old neighbor from across the street. Adaption is not only made to the role a person plays in our lives, but also to the emotional state somebody is in. When somebody is sad, we are trying to cheer that person up, when another person is excited, we try to calm him down. In the same way, the other person adapts to us. It is a bidirectional exchange of various signals that defines whether we can or cannot get along with somebody else.

As discussed earlier, interaction with the cell phone is very frequent and still increasing. So, in order to improve the interaction, the cell phone should act like a "real" person and adapt to our needs and habits. Although the following [18] is to be seen as a quick gaze into the future, most of the technology is already available today.

First of all, the phone has to get the opportunity to learn, so it analyzes our personality by evaluation of e.g. inter-event time, entropy of places visited or the amount of contacts interacted with regularly as described in chapter four, it gathers data regarding the location or, the personal information as described in second chapter and it has to get used to our habits such as checking the weather forecast in the morning, checking for the next train in order to get to our workplace or having a look at the current traffic situation in order to not be stuck in traffic jams. After the learning process, the cell phone could do all of this automatically in order to save its owner some time. When leaving the house in the morning, the cell phone could give advice whether to take the train or the subway in order to get to the workplace as all information is accessible for it.

However, it can do much more than just adapting to our habits. Our phone can become much more than just a phone - It may become a partner that guides us comfortably through each day. As devices like Google glass or Galaxy Gear are on the rise, this offers new opportunities for interaction on the base of big data. A watch-like gadget may easily be equipped with a heart rate sensor. If this sensor detects stress, the phone could play the favorite song of its owner in order to calm him down. Blood sugar levels could be tested as well and the person could be notified by the smartphone when anomalies are detected[7] which would be a blessing for people suffering from diabetes.

So, in the future even though we may have the same type of phone as a friend of ours, the phones may behave in a completely different way as they are adapted to personal needs. This development is already set in motion, the technology is ready and future is about to become reality.

5. UNIQUE IN THE CROWD

In this chapter, the central topic of the paper will be dealt with. Are our mobility traces unique? And, if they are, what exactly does that mean for us? Is there anything one can do about it?

5.1 Are we unique?

According to dictionaries, uniqueness describes the state of being the only one of its kind, without an equal or equivalent. [16] Every person is unique but is his phone data unique as well? Is it possible to determine which phone data corresponds with which person despite of anonymization of the data sets?

As described before, each cell phone reveals its location as soon as it is connected to a wireless network, the GSM network or the GPS function is activated. From now on, focus will be laid upon determining the location via tracking through the GSM network which allows an accuracy of up

doi: 10.2313/NET-2014-08-1_08

to 100 meters. By combining this information with the current time, spatio-temporal data is created. Together with all other kinds of information gathered, datasets are created. These datasets are simply anonymized - similar to the anonymity, carriers provide when they sell the data aggregated with other users to third parties. Hence, those sets do not contain any personal data such as name, personal address or other distinctive personal identifiers. So, what will be shown is the fact that without any former knowledge about the person, it is possible to gain important information.

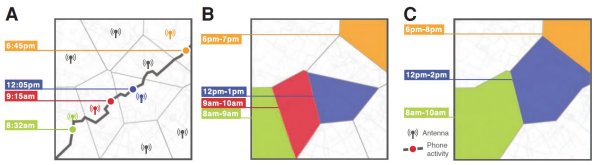


Figure 5: Mobility trace of a single dataset[1]

In Figure 5, the actual mobility trace is depicted in [A]. In [B] the position is depicted by analyzing the signal ranges of the base station that recorded the interaction. In [C] this information aggregated over longer time periods and base station's ranges are grouped.

What can we conclude out of the spatio-temporal data? When the location is tracked at 3am, it is most likely to be the person's home address while a location tracked at 10am is most likely to be that person's workplace. However, usually more than one person is living within a radius of 100 meters, same goes for workplaces. Hence, two points of spatio-temporal data is in most cases not enough in order to determine the phone user corresponding with the data.

So, in terms of mobility traces, this leads to the conclusion that we can only speak of a unique mobility trace when no other trace contains the same locations at the same time. Hence, a location traced at the Marienplatz in Munich at 5am is more likely to make a trace unique than the same place traced in the afternoon. With a population of slightly under 1.4 million, Munich is the third largest city of Germany. In 2013, an amount of 115 million mobile phone contracts were registered [21] in Germany. Assuming that mobile phone contracts are uniformly distributed in the population (and so, there are no variances due to regional aspects) that 1.75% of the contracts nationwide may be registered in Munich which leads to a total number of about 2 million contracts. As shown before in Figure 1, Munich is covered by an immense amount of GSM base stations, so the resolution of the antennas used is yery high. Is it possible to find unique mobility traces in a city as big as Munich?

It is, and it depends on the amount p of the spatio-temporal data extracted as well as on the cardinality of I which is the set that includes all spatio-temporal points. While those points are determined automatically on a regular basis, further interaction due to frequent phone calls or text messaging will increase the amount of points as shown in Figure 6. Hence, increased interaction at the same location will blur the result as the probability of randomly choosing a different location decreases. The higher p, the more increases the

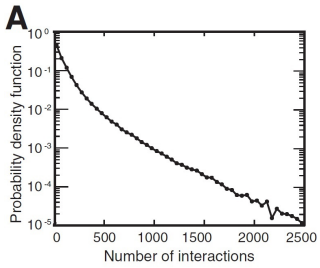


Figure 6: Probability Density with increasing amount of interactions[1]

probability that the mobility trace becomes unique. If the location at noon and at midnight is extracted, this most likely is not enough to find unique traces. By adding a third location to the set, the cardinality of the subset of traces, that still have the same locations at the same times decreases. It may decrease to a cardinality of 1 and $|S(Ip)| = 1$ means that the trace has become unique as shown in Figure 7A and B.

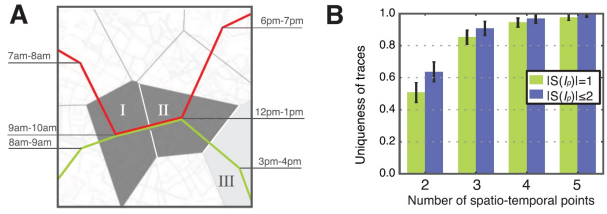


Figure 7: Uniqueness in respect to p[1]

A similar study [1] over 15 months with datasets of 1.5 million people has shown that four randomly chosen spatio-temporal points uniquely identify 95% of the traces while two points still identify more than 50%. By aggregating calls into three hour periods or even six hour periods and by grouping adjacent cell phone ranges, the uniqueness which is characterized by ϵ decreases. The uniqueness of a trace can be calculated by the function $\epsilon = \alpha - (vh)^{\beta}$ in which v is representing the resolution of antennas and h the resolution of hours. Hence, the output data is highly sensible to aggregation and traces at a high resolution are more likely to be unique. Statistically, uniqueness is higher when resolution of either antennas or time is coarse and the other one high. However, by modifying the estimators α and β, the uniqueness ϵ decreases by a constant factor. This can be compensated by increasing the amount p of points reviewed. If p is high enough, the uniqueness ϵ will be high enough despite of the coarse resolution.

So, it is possible to uniquely identify mobile traces. But is it also possible to link one to a specific person and so, to deanonymize the dataset? As for now, only the location data has been used. Adding the data from social networks, evaluating photos posted on Instagram, check-ins at restaurants and meetings with friends in "real life" will create a social graph which can be evaluated accordingly. A study held in 2012 [3] showed that 80% of the traces could successfully be linked while only 8% of the traces were linked incorrectly

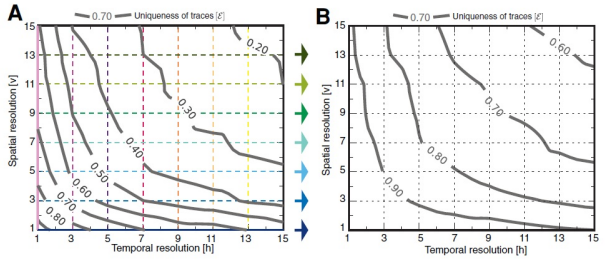


Figure 8: Uniqueness density with $p = 4$ (A) and $p = 10$ (B)[1]

while the remainder of the set could not be identified without a doubt. In this study, a contact graph based on the mobility trace is created. Additionally to that, a social graph consisting out of data that is publicly available at social networks as Facebook or LinkedIn is created. Those graphs are compared through various algorithms such as lowest edit distance[4]. If an accordance between the graphs is found, e.g. when two persons only have one mutual friend in a social network and meet regularly with each other and a third person in "real life", those traces would be very likely to be identified. The more vertices are identified, the easier it will be to identify the remainder, especially if the vertices are central and have many outgoing edges.

5.2 How can privacy be reclaimed?

But what does it mean for the individual when his anonymized mobility trace can not only be identified uniquely but also linked back to his personal data? It is a major threat! When the home address has been successfully linked to a mobility trace, it is a good target for thieves as they could find out easily during what hours the home owners are usually at work, school or at the gym. Frequent visits in a hospital could be interpreted as an indicator for bad health. If this information was accessible to an insurance company, monthly fees will either be a lot higher than usual or the person might even be rejected by the insurance company. These are only two of many possible scenarios of what might happen, if de-anonymized mobility traces were publicly available. They already make clear, that this must be prevented from happening. But how can this be prevented?

"Immediately throw away your smartphones!", some might postulate. This however is most certainly not a good solution as that can only be seen as a hysteric over-reaction. It is necessary though that information awareness is built up. When the user is in control of the data, he knows which information is shared and can administrate it accordingly.

In order to do so, insight about the data provided needs to be gained. Often, it is hard to retrace the information access granted in the telephone's operating system. Thus, programs like TaintDroid[8] were developed in order to visualize the sensitive information shared. This program does not only visualize the data but also determines whether an application is accessing information it is not supposed to. Knowing where data is leaked is only the first step. Actions must be taken in order to stop the leakage.

[4]Minimal change due to adding or removing vertices

There are many different models how personal data can be administrated by the phone user so he stays in control. One possibility proposed by the same researcher [4] who also conducted the study "Unique in the crowd" described before, is to setup personal data stores. Those were to be filled with data collected by the cell phone and given by the phone owner. If an application or different service needs personal information, the owner can grant the right to access information to the respective service. As the data storage is self-administrated, the user may delete personal data at any time and make his personal information inaccessible. This however, is only an idea how to solve this problem and would require a change of laws that prevents companies from saving or selling data. A change is not in sight, yet.

6. CONCLUSION

In this paper, I have described what kind of data is gathered by the usage of the cell phone and to whom it is accessible. With this information being processed for personality prediction and personality adaption, the window of opportunities for the smartphone of the future is wide open. However, every person has to decide for his- or herself, whether he or she wants to keep feeding the phone with valuable personal data. Is the product received valuable enough to weigh up the data provided? However, to isolate oneself from this development is a tilt at windmills. Being a socially connected individual will require you to give up some of your privacy. Being unique in the crowd, however, is a problem that will have to be solved in order to protect the individuals' privacies. The scenarios described must be prevented; so, the user needs to regain control which, however, cannot be achieved without the help of the respective government. It will be a struggle but every way begins with a first step... And if the whole crowd is moving, the persons involved are no longer unique but uniform.

7. REFERENCES

[1] Yves-Alexandre de Montjoye, César A. Hidalgo, Michel Verleysen and Vincent D. Blondel: *Unique in the Crowd: The privacy bounds of human mobility*, pages 1-5, PDF, Boston, MA, USA, 2012

[2] Yves-Alexandre de Montjoye, Jordi Quoidbach, Florent Robic and Alex Pentland: *Predicting people personality using novel mobile phone-based metrics*, pages 1-3, PDF, Boston, MA, USA, 2013

[3] Mudhakar Srivatsa and Mike Hicks: *Deanonymizing Mobility Traces: Using Social Networks as a Side-Channel*, pages 1-10, PDF, College Park, MD, USA, 2012

[4] Yves-Alexandre de Montjoye, Samuel S. Wang and Alex Pentland: *On the trusted use of Large-Scale Personal Data*, pages 1-4, PDF, Boston, MA, USA, 2012

[5] Atle Arnes and Catharina Nes: *What does your app know about you?*, Datatilsynet, pages 14-16, PDF, Oslo, Norway, 2011

[6] Timothy A. Judge, Chad A. Higgins, Carl J. Thoresen and Murray R. Barrick: *Big five personality traits, General Mental Ability, And Career Success across the life span*, Datatilsynet, pages 623-625, PDF, Iowa City, IA, USA, 1999

[7] Nuria Oliver and Fernando Flores-Mangas: *Health*

Gear: A Real-Time Wearable System for Monitoring and Analyzing Physiological Signals, Microsoft Research, pages 1-4, PDF, Redmond, WA, USA, 2006

[8] William Enck, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, Peter Gilbert, Byung-Gon Chun and Anmol N. Sheth: *TaintDroid: An Information-Flow Tracking System for Real-Time Privacy Monitoring on Smartphones*, USENIX Symposium, pages 1-15, PDF, University Park, PA, USA, 2010

[9] Chao Li, Daniel Yang Li, Gerome Miklau and Dan Suciu: *A Theory of Pricing Private Data*, University of Massachusetts, pages 1-12, PDF, Amherst, MA, USA, 2013

[10] *Global mobile statistics*, `http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats/a#subscribers`

[11] *Verizon selling personal data*, `http://money.cnn.com/2011/11/01/technology/-verizon_att_sprint_tmobile_privacy/`

[12] *Every second mobile phone is a smart phone*, `http://www.format.at/articles/1314/923/355857/das-handy`

[13] *NSA can activate microphone and camera*, `http://bgr.com/2013/12/31/nsa-iphone-hack/`

[14] *What Does My Cell Phone Carrier Know About Me?*, `http://bgr.com/2013/12/31/nsa-iphone-hack/`

[15] *What Does Google Know About You?*, `http://www.channel4.com/news/what-does-google-know-about-you`

[16] *Definition of uniqueness*, `http://www.thefreedictionary.com/uniqueness`

[17] *How Big Data profits from your personal information*, `http://www.theglobeandmail.com/globe-debate/follow-your-data-from-your-phone-to-the-marketplace/article17056305/`

[18] *Presentation My Cellphone - My Partner by Nuria Oliver*, `https://www.youtube.com/watch?v=kk-eJoK4fug`

[19] *Big data and social good*, `http://dynamicinsights.telefonica.com/1146/big-data-and-social-good-nuria-oliver`

[20] *Telecom firms mine for gold in big data despite privacy concerns*, `http://www.reuters.com/article/2014/02/23/us-mobile-world-bigdata-idUSBREA1M09F20140223`

[21] *Amount of cell phone contracts in Germany*, `http://de.statista.com/statistik/daten/studie/3907/-umfrage/mobilfunkanschluesse-in-deutschland/`

[22] *Mobile content usage*, `http://www.statista.com/statistics/187128/mobile-content-usage-in-the-us-by-share-of-subscribers/`

[23] *GSM base stations in Munich*, `http://maps.muenchen.de/rgu/mobilfunkstationen`

[24] *Track My Life*, `https://tml.me/`

[25] *Facebook likes predict personal traits*, `http://news.sciencemag.org/2013/03/facebook-preferences-predict-personality-traits`

doi: 10.2313/NET-2014-08-1_08

Survey: Security in Smart Building Networks

Michaela Mohl
Betreuer: Holger Kinkelin
Seminar Future Internet SS2014
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: mohl@in.tum.de

KURZFASSUNG

Heutzutage werden vor allem bedeutende, industrielle Gebäude und jene mit infrastruktureller Wichtigkeit immer weiter automatisiert. Das Ziel dieser Gebäudeautomatisierung ist es, durch Automatisierung Gebäude zu konstruieren die sicherer, ökologischer und komfortabler sind. Zu Begin des Einsatzes waren dies in der Regel geschlossene Autonome Systeme. Diese konnten nur durch direkt vor Ort manipuliert werden.

Durch die Verbreitung von Gebäudeautomationssystemen und des Internets hat sich der Trend dahingehend entwickelt, dass sogar Systeme dieser Art durchgehend vernetzt werden. Beispielsweise könnten verschiedene Gebäude einer Firma so verbunden werden, dass Fernzugriffe ermöglicht werden. Optional ist diese Verbindung durch Anschluss an das Internet realisierbar. Dadurch eröffnen sich jedoch Problemstellen und Gefahren, die das Netzwerk im Gegenzug zu den anfangs geschlossenen Systemen unsicherer machen.

Das Ziel dieser Arbeit ist es, die Sicherheitslücken zu präsentieren und Lösungsmöglichkeiten für den Schutz zu analysieren. Dies wird anhand einer Einführung in den Aufbau von Gebäudeautomationssystemen und in einen der Marktführer BACnet dargestellt. Um sich vor Angriffen schützen zu können, müssen die potentiellen Angreifer bekannt sein. Diese werden mit ihren Angriffsarten erörtert.
Zudem werden Sicherheitseigenschaften von verschiedenen Gebäudeautomationssystemen, BACnet, KNX und LonWorks, beschrieben und verglichen.
Im Rahmen der Arbeit wird sich herausstellen, dass BACnet seiner Konkurrenz im Punkt Sicherheit deutlich voraus ist und vor allem einen individuelleren Schutz ermöglicht.

Schlüsselworte

BACnet, Security, Smart Building Networks, Building Automation Systems, BAS, KNX, LonWorks, LonTalk

1. EINLEITUNG

Bedeutungsvolle Gebäude, wie zum Beispiel Flughäfen, nutzen seit längerem Gebäudeautomationssysteme. Früher waren diese Systeme nur direkt vor Ort an der Arbeitsstation manipulierbar. Illegaler Zugriff lies sich durch Restriktion authorisierter Personen einschränken.

Heutige Gebäudeautomationssysteme werden zunehmend vernetzt. Es könnten beispielsweise alle Systeme deutscher Flughäfen verbunden werden. Alternativ könnte auch ein Fernzugriff auf einen Flughafen ermöglicht werden.

Diese Verbindungen bergen die Gefahr des illegalen Zugriffes und der Manipulation, da die Menge der Angriffspunkte wesentlich angestiegen ist. Damals gab es nur den direkten Eingriff über die Arbeitsstation, mittlerweile hat sich dies durch die Internetanbindung vervielfacht.

Das Ausmaß eines solchen Angriffes kann enorme Schäden verursachen. Falls ein umfangreiches System in einem Flughafen angegriffen werden sollte, könnte der Angreifer beispielsweise den Feueralarm aktivieren und gleichzeitig alle automatischen Türen verriegeln. Dies könnte im schlechtesten Fall zu einer Massenpanik führen und Menschenleben gefährden.

Um die Problemfelder eines Gebäudeautomationssystems zu erkennen, muss dessen Struktur bekannt sein. Dies wird anfänglich veralgemeintert in Kapitel Zwei erklärt, danach in Kapitel Drei gibt es detallierte Erläuterung von dem Marktführer BACnet.

In Kapitel Vier werden die potentiellen Angreifer und Angriffsarten analysiert, zudem, in Kapitel Fünf und Sechs wird der Schutz, drei gängiger Gebäudeautomationssysteme gegen Angriffe betrachtet und verglichen.

2. SMART BUILDING NETWORKS

Das folgende Kapitel stellt eine Einführung in die Thematik dar. Beginnend wird der Begriff Smart Building Networks erläutert. Abschließend wird die Struktur von diesen dargestellt. Dies geschiet basiert auf den Quellen [1] und [2].

2.1 Smart Building Networks

Unter einem Smart Building Network oder auch der Gebäudeautomatisierung versteht man die netzwerkgestützte Regelung und Optimisierung von Heizung, Lüftung, Licht und Klimaanlage. Diese existieren in vor allem größeren und wichtigeren Industriegebäuden, können aber auch in kleineren Industrien oder sogar in Privathäusern vorgefunden werden, auch wenn sich der finanzielle Aufwand für letztere nur eingeschränkt lohnt.

2.1.1 Struktur eines Smart Building

Ein Gebäudeautomationssystem kann in drei logische Ebenen aufgeteilt werden. Hierbei wird zwischen Rolle und Verantwortlichkeit getrennt. Die folgende Erklärung erfolgt

doi: 10.2313/NET-2014-08-1_09

anhand der Grafik [1] und beginnt mit der untersten Ebene.

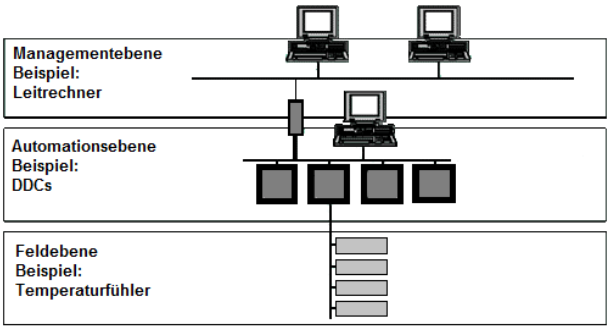


Abbildung 1: Die Ebenen [2]

- Feldebene

  Die Feldebene besteht aus simplen Sensoren, beispielsweise Temperaturmessgeräten, und Aktoren, wie Heizungsventilen. Diese sammeln Informationen, welche an die Automationsebene weiter geleitet wird. Sie können aber auch Befehle aus der darüberliegenden Ebene erhalten und diese ausführen. Beispielsweise kann auf oberen Ebenen der Befehl "Alle Lichter sollen um 19 Uhr ausgeschalten werden" implementiert sein, dies wird dann auf den unteren Ebenen ausgeführt.

- Automationsebene

  Bei der Automationsebene handelt es sich um eine Zwischenebene, die die Daten der Feldebene evaluiert und die verarbeiteten Informationen an die übergeordnete Ebene in Echtzeit weiterleitet. Mit den Direct-Digital-Controls (DDCs)[1] wird es ermöglicht, mehrere Feldgeräte zusammenzufassen, um komplexere Befehle auszuführen und mehr Daten zu sammeln, die miteinander einen höheren Informationsgehalt erzielen. Beispielsweise können an diesen DDC mehrere Lichtschalter angeschlossen werden, um einmalig die Lichter mit einem Befehl zu löschen. Auf dieser Ebene befindet sich in der Regel ein BUS, der den Leitrechner der übergeordneten Ebene mit den Direct-Digital-Controls aus dieser Ebene verbindet.

- Managementebene

  Die Managementebene ist übergeordnet und verantwortlich für die Betriebsführung, Meldeüberwachung und das Energiemanagement. Sie übernimmt eine organisierende Rolle. Beispielhaft könnte ein Gebäudeautomationssystem derart aufgebaut sein, dass ein oder mehrere Leitrechner übergeordnete Funktionen übernehmen, welche durch die Managementebene realisiert wird. Ein Leitrechner ist ein System, welches alle Informationen der untergeordneten Ebenen sammelt und reguliert. Dieses stellt eine Schnittstelle zur Interaktion mit Menschen dar.

[1] Direct-Digital-Controls sind Computerähnliche Module die eine Menge von Feldgeräte verbinden und Informationen in Echtzeit weiterleiten.

3. BACNET

Alle Informationen aus diesem Kapitel über BACnet entstammen den Quellen [1], [2], [3] und [4].

BACnet (***B**uilding **A**utomation and **C**ontrol **Net**works*) ist ein Datenkommunikationsprotokoll für Gebäudeautomatisierung und deren Kontrollnetzwerke.

Dieses Protokoll wurde von der ***A**merican **S**ociety of **H**eating, **R**efrigerating and **A**ir-Conditioning **E**ngineers* (ASHRAE) ab 1987 entwickelt. Ihr Ziel war ein Standardprotokoll für die Gebäudeautomatisierung zu etablieren, welche mit der BACnet ISO-Norm 16484-5 im Jahr 2003 erfolgreich umgesetzt wurde.

3.1 BACnet Konzept

Aufgrund der Notwendigkeit, die unterschiedlichen Komponenten in einem Gebäude mit einem standardisierten Kommunikationsprotokoll ansprechen zu können, entstand BACnet. Hierbei sollten Interoperabilität und Herstellerunabhängigkeit gewährleistet werden.

BACnet ist ein Gebäudeautomatisierungssystem, das sich problemlos in die drei gerade genannten Ebenen eingliedern lässt. Es hat übergeordnete Dienste, die es anbietet, und sogenannte Objekte, die Element des Systems sein können wie beispielsweise ein Feldgerät. Diese Dienste werden im Weiteren erklärt.

Für alle drei Ebenen kann BACnetals Software verwendet werden. Jedoch zeigt BACnet seine Stärken im Einsatz auf den beiden übergeordneten Ebenen, der Automationsebende und der Managementebene, weshalb es zum Großteil nur hier benutzt wird. In diesen Fällen kann BACnet mit einem anderen System, beispielsweise LonWorks oder KNX, auf der Feldebene zusammenarbeiten.

3.1.1 Dienste

BACnet bietet 38 Dienste an, die in die folgenden sechs Klassen eingeordnet werden können:

- Objektzugriff

  Diese Dienstklasse beinhaltet typische Ein- und Ausgabeoperationen wie Lesen und Schreiben. Außerdem sollen Objekte damit angelegt und gelöscht werden können.

- Device- und Netzwerkmanagement

  Dem Betreiber wird es durch diese Funktionen ermöglicht, einen automatischen Neustart mit Standardkonfigurationen des Netzwerks durchzuführen.

- Alarm- und Eventfunktionen

  Wenn ein BACnet Device Alarmzustände, Betriebszustände oder eine Form von Fehlerzustand erkennt, so werden diese von den Alarm- und Eventdiensten behandelt. Ein BACnet Gerät verfügt über drei Zustände: "*normal*", "*offnormal*" und "*fault*".

- Remote Device Management

  Dies ermöglicht Fernzugriffe auf verschiedene Eigenschaften. Beispielsweise kann ein Gerät damit neu

doi: 10.2313/NET-2014-08-1_09

gestartet werden. Darunter fallen noch zwei weitere spezielle Dienste, nämlich das *Dynamic Device Binding*, welches erlaubt, Geräte per Broadcast zu ermitteln, und das *Dynamic Object Binding*, das es ermöglicht, Objekte dynamisch zu ermitteln.

- Virtual Terminal

  Das Virtual Terminal wird eher selten verwendet. Dieses ist für das Nutzen einer virtuellen Schnittstelle des Gerätes geeignet.

- Netzwerk-Sicherheit

  Es können individuell Sichheitsmechanismen eingefügt werden, welche im Kapitel [5] "Sicherheit im BACnet" näher erläutert werden.

Wichtig sind diese Informationen zu den Diensten deshalb, da all diese eine Angriffsmöglichkeit auf das System bieten. Dieser Dienstmissbrauch muss unterbunden werden. Zum einen könnte das System direkt manipuliert werden, indem beispielsweise ein Objekt gelöscht wird. Zum anderen wäre ein nicht erkannter Eingriff möglich, der später die Fernzugriffsdienste ausnutzen könnte.

3.1.2 Schichten

BACnet nutzt das OSI-Modell als Referenzmodell. Auf den untersten zwei Schichten, der Bitübertragungs- und der Sicherungsschicht, wird BACnet nicht verwendet. Hier werden andere Protokolle, wie zum Beispiel LonTalk, eingesetzt.

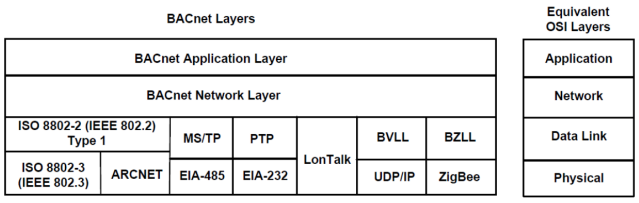


Abbildung 2: Die BACnet Schichten im Vergleich zum OSI Model [4]

Die Transport-, Sitzungs- und Darstellungsschicht, also Schicht 4-6 im OSI-Modell, beziehungsweise deren Funktionen sind, soweit erforderlich, in die BACnet-Anwendungsschicht mit integriert.

4. SICHERHEITSASPEKTE IN DER GEBÄUDEAUTOMATION UND BACNET

Es gibt zwei Gründe warum Gebäudeautomationssysteme grundsätzlich geschütz werden sollten.
Auf der einen Seite möchte man wertvolle Informationen versteckt halten, auf der anderen Seite sollen unterstützte Funktionen nicht missbraucht werden.

Das bedeutet, dass alle Informationen, über die das System verfügt, nur authorisierten Personen zur Verfügung stehen darf. Beispielhaft könnte ein Gebäudeautomationssystem mit einer Datenbank verbunden sein, die mit sensiblen Daten versehen ist oder gar selbst vertrauliche Daten sammelt. Diese sollten unter keinen Umständen von Dritten eingesehen oder abgehört werden.

Auch lässt sich erklären, wieso der Missbrauch von Funktionen vermieden werden muss. Das Eingangsbeispiel aus der Einleitung ist einleuchtend. Kein unbefugter Dritter darf Zugriff auf die Türenverriegelung und den Feueralarm eines Flughafens haben, da ansonsten eine Panik nach Belieben verursacht werden kann. Der Normalzustand des Flughafens wäre nicht mehr gewährleistet.

In diesem Kapitel werden die potentiellen Angreifer und die möglichen Angriffsarten, basierend auf den Quellen [5], [6] und [7], für ein Gebäudeautomationssystem diskutiert.

4.1 Die Angriffe

Um ein System dieser Art gegen unerwünschte Eingriffe von Außen zu schützen, muss bekannt sein, welche Angriffspunkte das System aufgrund ihrer Dienste anbietet und welche Arten von Angriffen, die damit einhergehen, existieren können.

4.1.1 Die Angreifer

Neben dem klassischen **Hacker** gibt es noch drei weitere Personengruppen, die in den Betracht kommen sollten, falls versucht wird, potentielle Angreifer eines Gebäudeautomationssystems zu klassifizieren.

- Kriminelle

  Hierunter lassen sich Terroristen oder Diebe beispielsweise aufzählen, die sich Zugang zum System verschaffen möchten, um ein poltisches Statement zu bewirken.

- Ehemalige Mitarbeiter

  Verschiedene Berichte [5] zeigen, dass verärgerte, ehemalige Mitarbeiter zu einer ernsthaften Bedrohung gehören. Das liegt vor allem daran, dass diese Personen in der Regel ein grundsätzliches Verständnis über das Systems besitzen, welches sie aus Frust im Affekt ausnutzen könnten, um Schaden anzurichten.

- Konkurrenz

  Die Überwachung eines Gebäudeautomationssystems der Konkurrenz könnte für Unternehmensforschung und Optimierung genutzt werden.

Generell lässt sich sagen, dass es wichtig ist, dass grundlegende Sicherheitsmechanismen existieren sollten. Da die Anzahl der Gefahrenquellen beachtlich ist, sollte das System zumindest in der Lage sein, Sicherheitsmechanismen im Nachhinein hinzuzufügen und auszuführen.

4.1.2 Die Attacken

Es existiert eine breite Menge von Attackarten, die im Folgenden diverenziert werden. Diese unterscheiden sich grundsätzlich nicht von klassichen Attacken, im Folgenden wird jedoch der Zusammenhang mit BACnet und Gebäuteautomationssystemen beschrieben.

- Passwort-Attacken

  Durch Raten oder durch Brute-Force wäre es möglich, den unauthorisierten Zugriff auf eine Arbeitsstation zu

doi: 10.2313/NET-2014-08-1_09

gewinnen. Arbeitsstationen in diesem Zusammenhang sind (Leit-)Rechner, die sich verteilt in den zwei oberen logischen Ebenen befinden können.
Aufgrund dieser Passwort-Attacken sollte in Gebäudeautomationssystemen den Nutzern ein Autorisationsdienst angeboten werden, der einen gesicherteren Zugang zum System ermöglicht.

- Denial of Service

  Eine Dienstverweigerung durch Überlastung des Systems existiert auch in der Gebäudeautomatisierung. Dies geschieht durch eine unüberwindbare Menge unnützer Nachrichten.
  BACnet und LonWorks sind relativ anfällig für diese Art von Angriffen, da beide Systeme mit invaliden oder leeren Nachrichten an die Grenze ihrer Kapazitäten gebracht werden können.

- Spoofing Attacken

  Es exisitiert eine Vielzahl von Möglichkeiten, wie innerhalb eines BACnet-Systems gespooft werden könnte. Unter dem Spoofing-Angriff wird von einer fälschlichen Kommunikation ausgegangen, indem der Angreifer sich dem Kommunikationspartner gegenüber als eine bekannte Identität ausgibt, um damit Informationen zu erhalten, die er unter anderen Umständen nicht gewinnen könnte. Nur die Authentifikation sicherer BACnet-Nachrichten kann dies unterbinden.

- Abhörattacken, Snooping und Port-Scanning

  Abhörattacken sind lohnenswert für Diebe, um beispielsweise Informationen über den Zustand der Sicherheitseinrichtungen des Gebäudes zu gewinnen, was bei einem physikalischen Einbruch in das Gebäude zum Vorteil der Einbrecher führen würde. BACnet ist aufgrund seiner Struktur anfällig gegenüber Attacken dieser Art, da die simpleren Geräte im Netzwerk kaum in der Lage sind, sich eigenständig zu schützen.

- Exploit-Attacken

  Das Ausnutzen von Fehlern in der Hardware oder der Software ist einer der geringfügigeren Sicherheitsproblemen, kann jedoch zu dauerhaftem Fehlverhalten des Systems führen. Dies ist eine mögliche Angriffsart, wobei diese eher selten zu fatalen Fehlern führt. Aufgrund dessen ist dies nicht Hauptaugenmerk bei der Verteidigung von Gebäudeautomationssystemen.

Die Menge potentieller Angriffsarten ist beachtlich. Somit muss ein passender Schutz für Systeme dieser Art gefunden werden. Allerdings kann trotz der Tatsache, dass all diese Angriffsarten bekannt sind, nicht einfach bereits existierende Schutzsoftware verwendet werden, da die Struktur der Gebäudeautomationsysteme stark von der klassischen Netzwerkstruktur abweicht.

Deshalb muss ein individueller Schutz gefunden werden. Bei diesem Schutz muss darauf geachtet werden, dass alle Elemente des Systems äquivalent geschützt werden und Veränderungen unproblematisch vollzogen werden können.

4.2 Sicherheitsanforderungen

Um überhaupt eine sichere Umgebung schaffen zu können, muss ein Konzept von Sicherheitsanforderungen existieren. Um diese Bedingungen zu erfüllen, müssen zahlreiche Sicherheitsmechanismen implementiert sein.
Einer der Hauptaspekte ist hierbei, dass die übertragenen Daten geschützt werden müssen. Außerdem muss dafür gesorgt werden, dass die Managementebene vor unauthorisiertem Zugriff gesichert ist. Das letztere kann durch bekannte Authentifikationalgorithmen geschehen.
Um sicherzustellen, dass Daten unverändert übertragen werden, gibt es drei grundlegende Prinzipien, die beachtet werden müssen.

- Vertraulichkeit

  Das Offenlegen vertraulicher Daten muss in jedem Fall vermieden werden.

- Datenintegrität

  Es muss sichergestellt werden, dass Daten nicht von unauthentifizierten Dritten verändert werden. Falls dies nicht garaniert werden kann, muss zumindest auf eine Veränderung hingewiesen werden.

- Datenverbindlichkeit

  Als letztes muss garantiert werden, dass die gesendeten Daten zum aktuellen Zeitpunkt gültig sind. Dritte dürfen weder in der Lage sein, Nachrichten zu injezieren, noch diese wiederholt auszugeben.

Das sind die allgemeinen Punkte, die zu beachten sind, wenn ein Gebäudenetzwerk gegen Angriffe geschützt werden soll. Im Folgenden wird detailliert dargestellt, welche Sicherheitsmechanismen BACnet implementiert. Dies wird darauffolgend mit alternativen Lösungen konkurrierender Systeme, hier KNX und LonWorks, verglichen.

5. SICHERHEIT IM BACNET

Nun werden,anhand der Quelle [8], die Maßnahmen beschrieben, die ASHRAE einrichtet und einrichten wird, um BACnet vor unerwünschten Zugriffen zu schützen.
Der Sicherheitsaspekt war bis vor kurzem noch nicht im Augenmerk der ASHRAE. Das liegt vor allem daran, dass Gebäudeautomationssysteme erst kürzlich miteinander oder mit dem Internet verbunden wurden. Diese Verbindungen können Risiken mit sich bringen.

Grundsätzlich sollten sowohl die Anlage durch eine Firewall, als auch das Wide Area Network durch ein Virtual Private Network geschützt sein.

Fraglich ist, ob dies ausreichend ist, denn wie gut das System vor Angriffen von außen geschützt ist, liegt an vielen Details. Diese wären die Firewall-Einstellungen, Einbruchserkennung und die Überwachung der drahtlosen Zugriffe auf das interne Netzwerk.
Falls eines der oben genannten Punkte eine Schwachstelle in der Verteidigung darstellt, ist es möglich, dass das gesamte System kompromitiert wird. Ein Angreifer könnte somit Daten sammeln oder gar die Kontrolle gewinnen und die Kommunikation im System stören.

doi: 10.2313/NET-2014-08-1_09

Mit sicheren BACnet-Nachrichten wäre das BACnet-Gerät des Angreifers nicht mehr zum Datensammeln geeignet, da Nachrichten verschlüsselt sind. Desweiteren kann die Kommunikation nicht gestört werden, da das angegriffene Objekt keine gültige Nachricht verfassen kann.

Um die Schwachstellen zu schützen und BACnet generell sicherer zu machen, hat das BACnet-Komitee eine Anzahl von **Zielen** bekannt gegeben.

- Schutz aller BACnet-Geräte

  Damit keine Schwachstelle im System existiert, muss jedes BACnet-Gerät gleichwertig geschützt sein. Das heißt, dass der Overhead, die Codemenge und die zu erwartetende Leistung minimal gehalten werden muss. Erst dadurch sind auch simplere Geräte in der Lage, diese Standards zu erfüllen.

- Art von Angriffen

  Das System sollte, je nach Bedarf, vor den folgenden Angriffsarten geschützt sein: Replay, Spoofing und Denial of Service.

- Gesicherte Kommunikation innerhalb des Netzes

  Interne Netzwerkkommunikation, beispielsweise Broadcast-Nachrichten, sollten geschützt werden. Dies wird durch Signaturen und Verschlüsselung erreicht. Es soll der Advanced Encryption Standard (AES) eingehalten werden.

- Grundkonzept BACnet

  BACnet zeichnet sich durch Interoperabilität aus. Dies soll trotz Veränderungen beibehalten werden. Desweiteren ist es wünschenswert, dass diese Veränderungen leicht zu implementieren und in der Zukunft noch austauschbar sind, also interoperabel.

Handelsübliche Lösungen wie IPsec oder Kerberos führen leider nicht zu befriedigenden Lösungen, da IPsec große Overheads besitzt und beide Ansätze keine Interoperabilität zulassen.
Somit muss eine maßgeschneiderte Lösung für Gebäudeautomationssysteme gefunden werden. Wie das gestaltet werden könnte, wird im Folgenden erläutert.

5.1 BACnet-Netzwerk-Sicherheits-Architektur

Im Folgenden wird erläutert wie der Marktührer BACnet strkturiert wurde um sich gegen Angriffe zu schützen.

Damit BACnet in der Lage ist, die oben genannten Ziele umzusetzten, müssen vier sicherheitsspezifische Fähigkeiten hizugefügt werden.

Die **Geräte-Authentifizierung** wird durch das Verwenden von Nachrichten-Signaturen erreicht.

Die **Datenkapselung** wird durch geteilte Schlüssel und das Verschlüsseln von Nachrichten bewerkstelligt.

Desweiteren soll mit Hilfe geteilter Schlüssel die **Nutzer-Authentifizierung** stattfinden.

Außerdem soll eine **Nutzer-Autorisierung** eingerichtet werden.

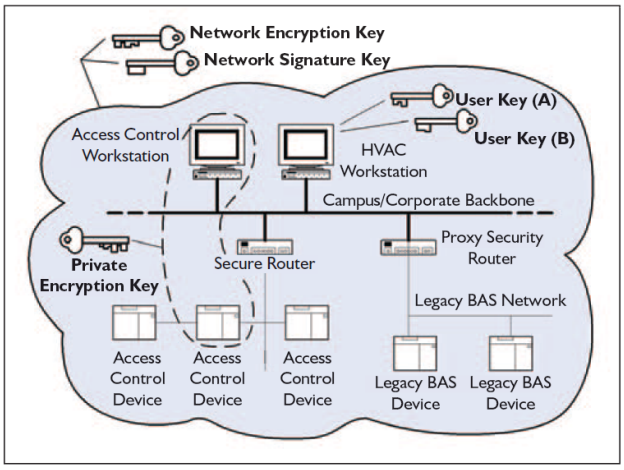


Abbildung 3: Die vier Schlüssel im BACnet [8]

Die oben gezeigte Grafik erläutert den Schlüsseleinsatz in BACnet. Die Managementebene wird hier als "Workstation" bezeichnet. Die Automationsebene wird mit dem BUS und den Routern dargestellt. Auf der Feldebene sind hier "Devices" verschiedenster Art aufgeführt. In BACnet werden vier verschiedene Schlüssel genutzt:

- Ein Signatur-Schlüssel, den sich alle Elemente in dem Netzwerk teilen. Dieser ist in der Skizze der "Network Signature Key".
- Ein Verschlüsselungs-Code, der allen im Netz bekannt ist. Dieser wird als "Network Encryption Key" bezeichnet.
- Es existieren private Verschlüsselungs-Codes, die nur eine Teilmenge der Netzwerkbeteiligten besitzen. In der Zeichnung ist ein Beispiel, wie bei der Kommunikation zwischen dem einen Access Control Device und der Access Control Workstation ein eigener "Private Encryption Key" verwendet wird.
- Als letztes gibt es noch Nutzer-Schlüssel, die eindeutig für eine Gruppe von Nutzern sind. Im Bild wären das der "User Key" A und B.

Zusammenfassend lässt sich über diese Schlüssel sagen, dass sie einen grundlegen Schutz bieten. Selbst mit dem Besitz von einem Schlüssel kann ein Eindringling nicht viel Informationen gewinnen beziehungsweise manipulieren.

Auch wenn der Eindringling den "Network Signature Key" erhält kann er nicht alle Nachrichten lesen. Da ein Teil der Nachrichten noch von einem "User Key" oder einem "Private Encryption Key" geschützt sind. Wenn jedoch ein illegaler Zugriff auf eine Arbeitsstation vorgefallen ist, kann eine Menge von Schlüsseln kompromitiert worden sein.

doi: 10.2313/NET-2014-08-1_09

5.2 Nachrichtenschutz

Da die Verschlüsselungsarchitektur nun erläutert wurde, wird anschließend noch erklärt, wie die versendeten Nachrichten verschlüsselt werden.

Im Folgenden wird beschrieben, wie ein grundlegender Schutz für BACnet-Nachrichten durch gewisse Techniken hinzugefügt werden kann.
Jede Nachricht soll mit Hilfe von HMAC und MD5 signiert werden. Außerdem sollen drei Identifikationsnummern enthalten sein, eine für die Quelle, eine für das Ziel und eine Nachrichten-ID. Zusätzlich soll noch ein Zeitstempel angehängt werden.
Durch diese zusätzlichen Informationen können bereits einfache unerwünschte Eingriffe vermieden oder zu mindest erkannt werden. Beispielsweise kann durch den Zeitstempel und die IDs Spoofing entdeckt oder sogar verhindert werden, jedoch müsste darauf geachtet werden, dass die Uhren der Geräte ungefähr gleich sind.

Weitere Schutzmechanismen können bei Bedarf hinzugefügt werden. Dies ist jedoch für diese Arbeit nicht von Belangen.

5.3 Nachrichten

Nun wird dargestellt, wie die Verschlüsselungen in eine Nachricht und den zugehörigen Header eingefügt werden. Zudem wird die Strukturierung der Nachricht erläutert.

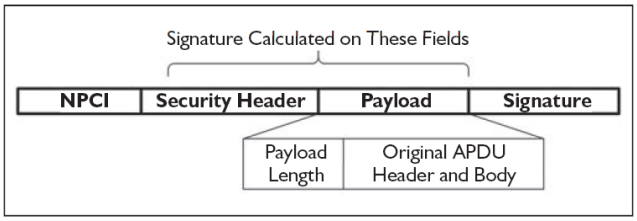


Abbildung 4: Ein Beispiel für die Sicherheitsvorkehrungen innerhalb einer Read Property Nachricht [8]

In der Abbildung 4 ist beispielsweise aufgeführt, wie eine Read Property Nachricht mit Sicherheitsvorkehrungen umhüllt wird.

Der Read Property Nachrichtenteil **A**pplication **P**rotocoll **D**ata **U**nit, ist der Inhalt der zu übertragenden Nachricht, welcher in die Payload platziert wird. Der Security Header gibt an, dass die APDU in der Payload enthalten ist.
Der eigentliche Nachrichten-Netzwerkschicht-Header wird zum **N**etwork **P**rotocol **C**ontrol **I**nformation aktualisiert, um darauf hinzustellen, dass diese Nachricht nun eine Netzwerkschicht-Nachricht ist.
Die Signatur wird mit Hilfe der Sicherheits-Header und der Payload berechnet und der Nachricht angehängt.

5.4 Nutzer-Autorisierung

Für die Nutzer-Autorisierung soll ein einfacher Mechanismus eingefügt werden. Dieser Mechanismus bietet ein 16-Bit-Nutzer-ID-Feld, das einen Nutzer identifiziert, welcher eine Operation anfragt. Desweiteren soll eine Nutzer-ID-Challenge-Sicherheitsnachricht existieren, die die gewünschte Nutzer-ID verifiziert.

Dies bietet den einfachsten Schutz und sollte in der Zukunft noch weiter ausgebaut werden.

5.5 Einordnung BACnet

Generell kann man über BACnet urteilen, dass es schon relativ wichtige Zwischenziele im Punkt Sicherheit erreicht hat. Vor allem gegen Spoofing- und Denial of Service-Angriffe ist BACnet gut geschützt. Jedoch hat BACnet ein Problem bei der Verteidigung. Ein Leitrechner in der Abbildung 3, beispielsweise die Access Control Workstation oder die HVAC [2] Workstation, besitzen in der Regel eine höhere Anzahl von Schlüsseln. Sollte also ein Gerät dieser Art angegriffen worden sein, sind somit alle Schlüssel, die dieses Gerät besitzt, zugänglich. Daher müssen die Leitrechner besonders stark geschützt werden.

Im folgenden wird diskutiert wie die Konkurrenz von BACnet mit diesen neuen Gefahren umgeht.

6. ALTERNATIVEN ZU BACNET

BACnet besitzt zwei Hauptkonkurrenzsysteme: LonWorks und KNX.
Wie sich diese im Aspekt der Sicherheit zu BACnet unterscheiden wird in diesem Kaptiel anhand den Quellen [7] differenziert.

6.1 LonWorks

Die **L**ocal **O**perating **N**etwork Technologie der Firma Echelon ist ein genormtes Bussystem. Die eingesetzten Geräte besitzen eine eigene Intelligenz und kommunizieren mitteinander in einem lokalen Netz. Diese Dezentralisierung ist der Unterschied zum BACnet.

LonWorks bietet einen vierstufigen Challenge-Response-Authentifizierungsmechanismus.
Ein Sender, der den Wunsch hat, eine Übertragung authentifizieren zu lassen, bestätigt die Authentifizierungsbits seiner Nachricht. Der Empfänger antwortet mit einer 64-Bit Zufallszahl. Die Antwort des Senders darauf ist ein 64-Bit Hashwert. Dieser wurde mit Hilfe der vorherigen Zufallszahl und eines geteilten Schlüssels gehasht. Der Empfänger kontrolliert den erhaltenen Wert. Desweiteren wird die Identität des Nutzers verifiziert und die Daten werden auf Aktualität und Integrität geprüft.

Jedoch hat LonWorks die folgenden **Sicherheitslücken**:

- Da die Daten unverschlüsselt übertragen werden, kann keine Vertraulichkeit sichergestellt werden.
- Bei der Authentifizierung wird nur der Sender überprüft, nicht der Empfänger. Dies ermöglicht Spoofing vom Empfänger.
- Jeder Knoten kann nur einen Schlüssel besitzen. Somit müssen alle Knoten, die miteinander kommunizieren

[2] HVAC steht für Heating, Ventilation and Air Conditioning

doi: 10.2313/NET-2014-08-1_09

wollen, denselben Schlüssel verwenden. Im schlechtesten Fall ist durch ein kompromitiertes Gerät jedes weitere Gerät ansprechbar.

- Es ist nicht möglich, Kommunikationsblöcke zu etablieren. Für jede einzelne Datenübertragung ist es also nötig, die vier Challenge-Response-Authentifizierungsnachrichten zu senden.
- Bei einem Multicast würde jeder Empfänger eine eigene Zufallszahl generieren. Der Sender wäre verpflichtet, jede dieser Zahlen zu hashen, was jedoch einen hoher Rechenaufwand bedeutet. Ausgenutzt werden kann dieser hohe Rechenaufwand mit Denial of Service-Attacken.

Zusammenfassend kann man urteilen, dass LonWorks mit den Challenge-Response-Verfahren einen grundlegenden Schutz bietet. Dennoch kann dieses System genau gegenteilig ausgenutzt werden, da es anfällig für Denial of Service-Angriffe ist.

6.2 KNX

KNX ist ein industrielles Kommunikationssystem. Es wird zur informationstechnischen Vernetzung von Geräten verwendet. Somit ist KNX auf der Feldebene einzuordnen und nicht komplett vergleichbar mit BACnet, welches auch übergeordnete Dienste anbietet.

KNX bietet keine Mechnismen, um die Datenintegrität, Datenaktualität oder die Vertraulichkeit zu wahren. Es existiert auch kein Authentifikationsdienst. Geboten wird nur ein simples Zugriffskontrollschemata basierend auf Klartextpasswörtern.

Aufgrund der fundamentalen Struktur von KNX gibt es noch zwei **Sicherheitslücken**:

Es sind keine parallelen Verbindungen möglich. Wenn also eine Verbindung mit zwei Geräten aufgebaut ist, ignorieren diese alle weiteren Anfragen. Somit sind Denial of Service-Angriffe leicht möglich.
Desweiteren können Nachrichten leicht injeziert werden, da die Quellen versendeter Nachrichten gespoofed werden können.

Somit hat KNX den schwächsten Sicherheitsschutz im Vergleich zu den drei erwähnten Systemen.

7. FAZIT

Manche Gebäudeautomationssysteme müssen geschützt werden. Daher gibt es die drei grundlegenden Sicherheitsziele und daraus abgeleitete Sichheitsmechanismen. Diese wurden im BACnet erfolgreich umgesetzt.

BACnet bietet einen Schutz, der eine Teilmenge von Angriffsarten abwehrt. Der Schutz ist je nach Bedarf individuell anpassbar, skallierbar und aktualisierbar. LonWorks und KNX bieten weitaus weniger, wobei KNX hierbei das Schlusslicht bildet. Somit ist BACnet die geeignetste Wahl für Gebäudeautomationssysteme, die vor jeglicher Art von Angriffen geschützt sein müssen. Allerdings sind nicht alle drei Systeme in ihren Einsatzgebieten identisch und daher auch nicht komplett vergleichbar.

LonWorks und KNX müssen in der Zukunft noch ihre Lücken schließen. Vor allem in dem Kombinierten Einsatz mit BACnet muss darauf geachet werden das die Verbindung der Schwachstellen nicht zu zu vielen unschützbaren Punkten führt. Jedoch sollten alle Systeme in der Zukunft weiter den Sicherheitsaspekt im Auge behalten und austauschbare Sicherheitsmechanismen implementieren, da die Gefahr von Angriffen auch in der Zukunft gegeben sein wird und die Anzahl der Angriffen steigen könnte. Desweiteren können die Gebäudeautomationssysteme dann womöglich in tiefere Subssysteme reichen, die noch schützenswerterer sind als die klassischen Elemente der Gebäudeautomatisierung wie Heizung und Licht.

8. LITERATUR

[1] Gebäudeautomation Kommunikationssysteme mit EIB/KNX, LON und BACnet 2., neu bearbeitete Auflage, *Hermann Merz, Thomas Hansemann, Christof Hübner* Carl Hanser Verlag München 2009

[2] *ASHRAE BACnet Interest Group Europe*, `http://www.big-eu.org/bacnet/basics.php`

[3] *BACnet Dominik Ebert*, `https://prof.hti.bfh.ch/uploads/media/BACnet.pdf`

[4] *MBS Software BACnet Kurzübersicht*, `http://www.mbs-software.de/produkte/bacnet/`

[5] D. G. Holmberg, Ph.D., Member ASHRAE: *Enemies at The Gates Securing the BACnet® Building* BACnet® Today A Supplement to ASHRAE Journal, 2003

[6] C. Eckert *IT-Sicherheit*, Oldenbourg-Verlag, 2013, 8. Auflage

[7] Wolfgang Granzer, Wolfgang Kastner, Georg Neugschwandtner and Fritz Praus, *Security in Network Building Automation Systems* Wien

[8] D. G. Holmberg, Ph.D., Member ASHRAE: *Secure Messaging In BACnet®* BACnet® Today A Supplement to ASHRAE Journal, 2005

doi: 10.2313/NET-2014-08-1_09

Determinism for Ethernet flows in industrial networks

Lukas Grillmayer
Advisor: Fabien Geyer
Seminar Innovative Internet Technologies and Mobile Communications (IITM) SS2014
Chair for Network Architectures and Services
Department of Informatics, Technische Universität München
Email: lukas.grillmayer@tum.de

ABSTRACT

Due to technological advances, network architectures must handle a growing amount of exchanged data and number of connections. This results in increasingly complex systems. In the domain of industrial systems, it is necessary to provide performance guarantees for large networks. This includes bounded end-to-end latency, bounded end-to-end jitter, and maximum buffer usage.

This paper presents the analytic approach of Network Calculus by providing a basic theoretical background and applying it to simple systems. Finally it is shown how Network Calculus can be applied to Ethernet and AFDX (Avionics Full Duplex Switched Ethernet) networks.

Keywords

Determinism, Industrial Networks, Network Calculus, Ethernet, AFDX

1. INTRODUCTION

Networks enable communication between numerous computational devices and are omnipresent in today's world. As technology advances the amount of electronic communication partners and quantity of exchanged data increases drastically. This results in new challenges and requirements for a network infrastructure as the competition for network resources increases. Within the scope of this paper a network is considered to be a system which can be divided into sub systems such as servers or buffers. The effects of lacking resources are easily observable in the domain of real-time applications, where computer science differentiates between soft and hard systems. For example VoIP internet telephony is considered a soft system as a loss of frames will only result in degraded service quality, even system failure does not impose a major problem. In contrast, an aircraft autopilot is a hard real-time system. If a deadline is missed and sensor data arrives too late at the autopilot, the entire system might fail and endanger human lives as well as the physical environment [8].

Industrial networks often deploy custom software and hardware to achieve compliance with a set of requirements, needed for hard real-time applications. In other words, industrial networks must be deterministic. In order to provide guarantees, these networks need to be analyzed towards their worst-case behavior. Common approaches are Network Calculus, queuing networks simulation, and model checking [9].

This paper uses deterministic Network Calculus as described by Jean-Yves Le Boudec [3], who also provided two short tutorials on network calculus [1, 2].

The following section presents Network Calculus, a collection of theorems which forms a tool to analytically determine performance bounds for systems. This section introduces fundamental properties of networks and provides background information of the proposed mathematical framework. Section 3 discusses how Ethernet technology can be adapted to suit hard real-time applications and shows how Network Calculus can be applied to Avionics Full Duplex Switched Ethernet (AFDX).

2. NETWORK CALCULUS

Network calculus is an algebraic framework to describe and analyze fundamental properties of integrated service networks. It provides insight into network flow problems, to form an understanding of e.g. window flow control, scheduling, and buffer dimensioning. Using network calculus a system can be analyzed towards compliance to given performance constraints, i.e. required for real-time applications [3].

2.1 Basics

In contrast to traditional system theory network calculus operates on the Min-Plus algebra (also referred to as Min-Plus dioid, Min-Plus semiring or $< min, + >$) where addition is the computation of the minimum and multiplication is addition [1].

Convolution and deconvolution of two functions f and g are defined as follows [1]:

$$(f \otimes g)(t) = \inf_{0 \leq s \leq t} \{f(t-s) + g(s)\} \tag{1}$$

$$(f \oslash g)(t) = \sup_{u \leq 0} \{f(t+u) - g(u)\} \tag{2}$$

A data flow in a system is measured in data units per time interval, e.g. bit/s. It is characterized by a non-decreasing function $R(t) \in \mathcal{F}$, whereas $t \in \mathbb{R}^+$ denotes time [1, 4]:

$$\mathcal{F} = \left\{ f : \mathbb{R}^+ \to \mathbb{R}^+, \forall t \geq s : f(t) \geq f(s), f(0) = 0 \right\} \tag{3}$$

As an entire network is hard to model, it can be abstracted as a system consisting of numerous smaller subsystems, which are easier to analyze. Every system has an input flow $R(t)$, and an output flow $R^*(t)$. While processing data it imposes

doi: 10.2313/NET-2014-08-1_10

a (variable) delay $d(t)$ on an input flow. The amount of data traversing a system at time t is called backlog $R(t) - R^*(t)$. For a loss-less system this is defined as follows whereas τ denotes the time window width [4, 3]:

$$d(t) = \inf\{\tau \geq 0 : R(t) \leq R^*(t+\tau)\} \tag{4}$$

An example for a simple system $\mathcal{S}$ is a FIFO queue (or buffer), which experiences a constant input rate r, thus the input flow is characterized by $R(t) = rt$. The output flow is denoted as $R^*(t) = R(t + d(t))$. Figure 1 illustrates this with $r = 10Mbit/s$ and $d(t) = 5s$.

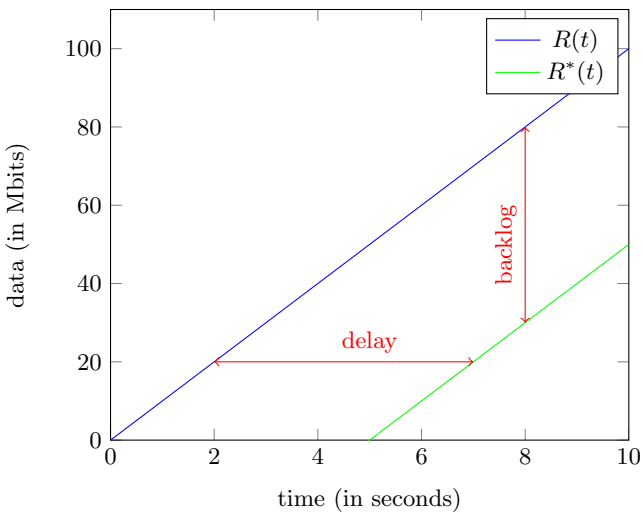


Figure 1: Illustration of input and output flow

2.2 Arrival Curves

Requirements for industrial networks commonly ask for guarantees for data flows. For a data flow $R(t)$ this can be modeled by an arrival curve $\alpha(t) \in \mathcal{F}$ with $t \geq 0$. $R(t)$ is limited by $\alpha(t)$ if for all $s \leq t$ [3]:

$$R(t) - R(s) \leq \alpha(t - s) \tag{5}$$

This is equal to:

$$R(t) \leq \inf_{0 \leq s \leq t} \{\alpha(t - s) + R(s)\} = (\alpha \otimes R)(t) \tag{6}$$

For example let $R(t)$ have an arrival curve of $\alpha(t) = rt$ then the flow is "peak rate limited" by $r\tau$ for any time window τ. In case of a constant arrival curve $\alpha(t) = b$ the flow will not accept more than b data ever.

As this representation assumes a loss-less system, affine arrival curves are commonly used to model the concept of leaky buckets. In this analogy a flow is represented as a bucket, which is leaking fluid (data) at a constant rate when not empty. Data is poured into the bucket. If additional data causes the bucket to overflow, the excess data is marked as "non-conformant" and will be discarded or buffered. An affine arrival curve $\gamma_{r,b}$ with burst tolerance (maximal number of parallel incoming bits) b and rate r is defined as follows [1]:

$$\gamma_{r,b} = \begin{cases} rt + b & t > 0 \\ 0 & t \leq 0 \end{cases} \tag{7}$$

In the domain of Integrated services framework of the Internet (Interserv) additional parameters for maximum packet size M and sustainable rate r are introduced to calculate the arrival curve. As a result the traffic specification (T-SPEC) can be given as (p, M, r, b) [1]:

$$\alpha(t) = \min\{M + pt, rt + b\} = \gamma_{p,M}(t) \wedge \gamma_{r,b}(t) \tag{8}$$

2.3 Service Curves

In addition to arrival curves, which have put constraints on a flow, systems must also offer guarantees to a flow. Generally this is realized by packet schedulers. Network calculus represents schedulers as service curves. A system $\mathcal{S}$ with input flow $R(t)$ and output flow $R^*(t)$ offers a service curve β to the flow if $\forall t \geq 0\ \exists t_0 \geq 0$ with $t_0 \leq t$ to satisfy the following [1]:

$$R^*(t) - R(t_0) \geq \beta(t - t_0) \tag{9}$$

This is equal to:

$$R^*(t) \geq \inf_{0 \leq s \leq t} \{\beta(t - s) + R(s)\} = (\beta \otimes R)(t) \tag{10}$$

2.4 Bounds for simple networks

Service and arrival curves can be used to simplify some expressions. For lossless networks with service guarantees following theorems can be applied to analyze a system towards guarantees concerning bounded end-to-end latency, bounded end-to-end jitter, and maximum buffer usage [3].

2.4.1 *Backlog Bound*

Theorem 1.4.1 (Backlog Bound): Assume a flow, constrained by arrival curve α, traverses a system that offers a service curve of β. The backlog $R(t) - R^*(t)$ for all t satisfies:

$$R(t) - R^*(t) \leq \sup_{0 \leq s \leq t} [R(t) - R(t - s) + \beta(s)] \tag{11}$$

$$\leq \sup_{0 \leq s \leq t} [\alpha(s) + \beta(t - s)] \tag{12}$$

2.4.2 *Delay Bound*

Theorem 1.4.2 (Delay Bound): Assume a flow, constrained by arrival curve α, traverses a system that offers a service curve of β. The virtual delay $d(t)$ for all t satisfies:

$$d(t) \leq h(\alpha, \beta) \tag{13}$$

2.4.3 *Output Flow*

Theorem 1.4.3 (Output Flow): Assume a flow, constrained by arrival curve α, traverses a system that offers a service curve of β. The output flow is constrained by the arrival curve

$$\alpha^* = \alpha \otimes \beta \tag{14}$$

doi: 10.2313/NET-2014-08-1_10

2.5 Applied Network Calculus

This section shows how the network calculus methods presented above can be applied to a small network.

2.5.1 Example 1 - Single System

Consider a network consisting of one server $\mathcal{S}$ with service curve $\beta_{10,10}$, and one traversing flow with arrival curve $\gamma_{5,25}$.

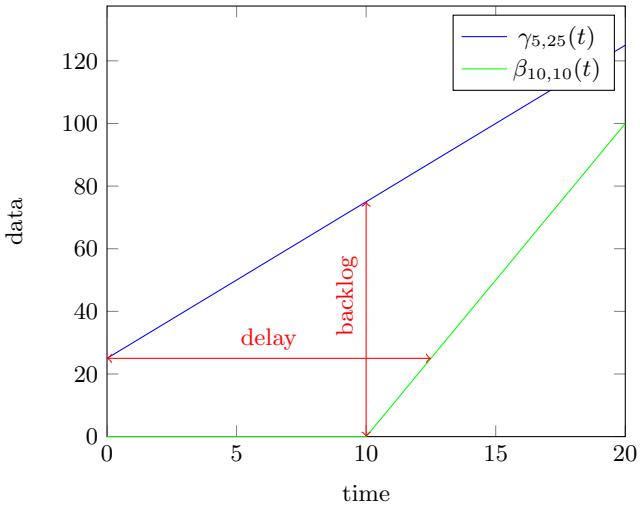


Figure 2: Arrival and service curve of a system

Figure 2 shows the system's arrival curve and service curve. The required backlog (or buffer size) of the server can be computed as presented in subsection 2.4.1. For all t the backlog bound must satisfy:

$$\begin{aligned} R(t) - R^*(t) &\leq \sup_{0 \leq s \leq t} [R(t) - R(t-s) + \beta(s)] \\ &\leq \sup_{0 \leq s \leq t} [\alpha(s) + \beta(t-s)] \\ &\leq \sup_{s \geq 0} [\alpha(s) - \beta(s)] \\ &\leq \sup_{s \geq 0} [\gamma_{5,25}(s) - \beta_{10,10}(s)] = 75 \end{aligned}$$

The maximum delay (or latency) of the flow can be computed according to subsection 2.4.2. For all t the delay bound must satisfy:

$$\begin{aligned} d(t) &= \inf \{\tau \geq 0 : \alpha(t) \leq \beta(t+\tau)\} \\ &= \inf \{\tau \geq 0 : \gamma_{5,25}(t) \leq \beta_{10,10}(t+\tau)\} \\ &\leq h(\alpha, \beta) \\ &\leq h(\gamma_{5,25}, \beta_{10,10}) \\ &\leq \sup_t \{d(t)\} = 12.5 \end{aligned}$$

Following equation 5 the constraint of the output flow (arrival curve) is as follows. For all t the arrival curve $\alpha^*(t)$ must satisfy:

$$\begin{aligned} \alpha^*(t-s) &\geq R^*(t) - R^*(s) \\ &\geq R^*(t) + 75 = 5t + 75 \end{aligned}$$

2.5.2 Example 2 - Concatenated Systems

Consider a network consisting of two servers $\mathcal{S}_1$ and $\mathcal{S}_2$ with service curve $\beta_{10,10}$, and one flow with arrival curve $\gamma_{5,25}$ traversing first $\mathcal{S}_1$ and then $\mathcal{S}_2$.

Some results from subsection 2.5.1 can be reused as the system $\mathcal{S}$ is equal to $\mathcal{S}_1$ now. Thus, the curve of the flow $R_1(t)$ after it has traversed $\mathcal{S}_1$ is

$$R_1(t) = 5t + 75$$

The curve of the flow after it has traversed $\mathcal{S}_2$, that is the entire system $\mathcal{S}$:

$$R^*(t) = R_1(t) + 75 = 5t + 150$$

The latency of the flow after $\mathcal{S}_1$:

$$\begin{aligned} D_1 &= \frac{b}{R_1} + T_1 \\ &= \frac{25}{10} + 10 = 12.5 \end{aligned}$$

The latency of the flow after $S2$:

$$\begin{aligned} D_2 &= \frac{b + rT_1}{R_2} + T_2 \\ &= \frac{25 + 5 * 10}{10} + 10 = 17.5 \end{aligned}$$

The resulting end-to-end delay D:

$$\begin{aligned} D &= D_1 + D_2 \\ &= 12.5 + 17.5 = 30 \end{aligned}$$

According to the *Pay Bursts Only Once* theorem [3] the end-to-end delay D_0 of concatenated systems can be improved as follows. Let $R = \min_i(R_i)$ and $T_0 = \sum_i T_i$:

$$\begin{aligned} D_0 &= \frac{b}{R} + T_0 \\ &= \frac{25}{10} + 20 = 22.5 \end{aligned}$$

3. ETHERNET

Ethernet was first standardized in 1983 as IEEE 802.3 and is the most wide spread transmission technology for local area networks today.

In early Ethernet networks a single coaxial cable was used as shared broadcast media in a bus architecture. If a connected host wanted to send a message, it had to wait until the medium is idle. All hosts were part of the same collision domain and if two hosts sent messages at the same time a collision occurred which was detected. After a suitable back-off strategy a host may retransmit the message. This strategy is referred to as CSMA/CD (Carrier Sense, Multiple Access, Collision Detection). At this time Ethernet was not suited for industrial networks as there was no central coordination for broadcast media access and the back-off interval increased exponentially plus a randomly chosen number to time slots, which results in non-deterministic behavior [7].

Over the years the coaxial cables were replaced by twisted pair copper cables. Common category 5 UTP cables include two twisted pairs of copper wire (Tx and Rx). In half-duplex

doi: 10.2313/NET-2014-08-1_10

mode the issues discussed in previous paragraph are not resolved since transmissions can still collide and an exponential back-off algorithm can result in very large transmission delays [7].

The usage of switches reduced the size of collision domains significantly. In Full-duplex Mode Ethernet there are dedicated wires for each directions of transmission, thus allowing simultaneous transmitting and receiving without collisions. Figure 3 shows an example for full-duplex, switched Ethernet, in which collisions are impossible but other issues need to be considered. In theory the Rx and Tx buffers can overflow, which results in packet loss. Using Network Calculus the buffers can be sized adequately to form a loss-less system. Furthermore, packets may be delayed due to congestion within the switch, which results in an unpredictable amount of jitter [7].

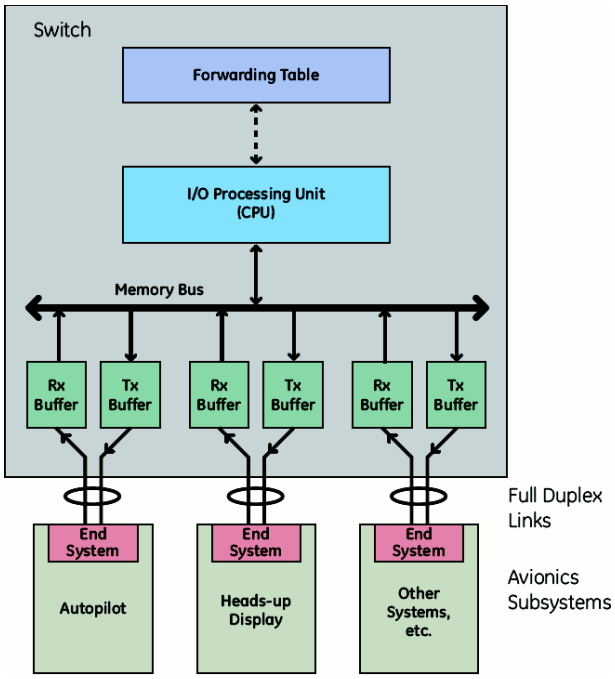


Figure 3: Full-Duplex Switched Ethernet [7]

In order to apply Network Calculus to actual hardware a model needs to be derived. To model an Ethernet switch we need to split the switch into smaller subsystems, which represent various functionality aspects of an Ethernet switch.

Being overly simplistic we are assuming omniscient knowledge about the network. Now, an unmanaged full-duplex Ethernet switch with sufficient backplane bandwidth could be modeled as a single system which adds a (variable) delay to an input signal. This is the time required to map the destination MAC address to a port and redirect the Ethernet packet correctly. Of course, this could also be split into more subsystems. The arrival curve would be determined by the maximum network speed and the maximum Ethernet packet size.

3.1 AFDX

In the domain of avionic networks, the Airbus Group developed an Ethernet derivative based on IEEE 802.3 called Avionics Full-Duplex Switched Ethernet (AFDX), which is defined in ARINC 664 section 7 [6, 7]. An example of AFDX deployment is the Airbus A380. The major difference to regular Ethernet is the usage of Virtual Links (VL). Portions of the available bandwidth of the physical link are dedicated to Virtual Links as illustrated in figure 4. Virtual Links are unidirectional fixed communication paths from one end system to one or more end systems.

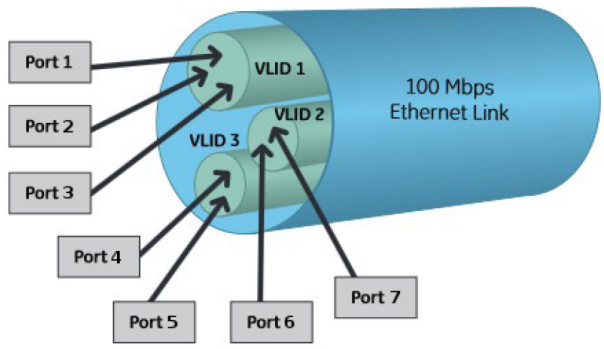


Figure 4: Three Virtual Links in a physical link [7]

For every VL the maximum packet size s_{min} and minimum time between two packets BAG is known. This allows for the formulation of guarantees by means of network calculus [6, 7]. Exemplary, an arrival curve $\alpha(t)$ for AFDX is defined as follows:

$$\alpha(t) = \min\left\{t, s_{min} + \frac{s_{min}}{BAG}t\right\}$$

4. TOOL: DISCO DNC

There are only few tools available for network calculus. The interested reader might want to look at the *DISCO Deterministic Network Calculator*. This Java library supports network analysis by means of deterministic network calculus presented above [5].

5. CONCLUSION

In the domain of industrial networks it is important to formulate performance guarantees. In addition to hardware constraints this requires the usage to deterministic protocols. As the development and certification of custom application specific standards is very expensive a current trend is to adapt Ethernet to fit the needs. An example for an Ethernet based derivative is AFDX, which adds central coordination by dedicating network bandwidth to certain communication paths. Using Network Calculus upper bounds can be calculated for required buffer size, end-to-end delay, and end-to-end jitter. It is apparent that upper bounds can only exist if the underlying model of the system has deterministic characteristics. Using traffic shapers even non-deterministic aspects can be modeled in an abstract manner to a certain extend although this might decrease the significance of the result. Network Calculus should only be used to analyze or model a system with hard requirements. The presented approach might be overly pessimistic as only the worst-case behavior is considered though it might be very unlikely. In

doi: 10.2313/NET-2014-08-1_10

order to evaluate the average system performance one should resort to network simulations.

6. REFERENCES

[1] J.-Y. Le Boudec, P. Thiran: *A SHORT TUTORIAL ON NETWORK CALCULUS - I: FUNDAMENTAL BOUNDS IN COMMUNICATION NETWORKS*, In IEEE International Symposium on Circuits and Systems, pages IV-93 - IV95, IEEE, Lausanne, Switzerland, 2000

[2] J.-Y. Le Boudec, P. Thiran, S. Giordano: *A SHORT TUTORIAL ON NETWORK CALCULUS - II: MIN-PLUS SYSTEM THEORY APPLIED TO COMMUNICATION NETWORKS*, IEEE, Lausanne, Switzerland, 2000

[3] J.-Y. Le Boudec, P. Thiran: *Network Calculus: A Theory of Deterministic Queuing Systems for the internet*, Springer, Berlin, Germany, 2001

[4] J.-P. Georges, T. Divoux, É. Rondeau: *Network Calculus: Application to switched real-time networking*, In Proceedings of the 5th International ICST Conference on Performance Evaluation Methodologies and Tools, pages 399-407, ICST, Vandœuvre lès Nancy, France, 2011

[5] *DISCO Deterministic Network Calculator*, Technische Universität Kaiserslautern, `http://disco.informatik.uni-kl.de/index.php/projects/disco-dnc` (accessed on 2014/07/01)

[6] C. Fuchs: *The Evolution of Avionics Networks From ARINC 429 to AFDX*, In Proceedings of the Seminars Future Internet (FI), Innovative Internet Technologies and Mobile Communication (IITM) and Aerospace Networks (AN), Summer Semester 2012, p.65-76, Technische Universität München, Munich, Germany, 2012

[7] General Electric: *AFDX/ARINC 664 Protocol: Tutorial*, General Electric, Fairfield, CT, USA

[8] Dr. C. Buckl: *Echtzeitsysteme WS 2013/14*, Technische Universität München, Munich, Germany, 2013, `http://www6.in.tum.de/Main/TeachingWs20123chtzeitsysteme` (accessed on 2014/07/01)

[9] H. Charara, J.-L. Scharbarg, J. Ermont, C. Fraboul: *Methods for bounding end-to-end delays on an AFDX network*, In Proceedings of the 18th Euromicro Conference on Real-Time Systems (ECRTS'06), Toulouse, France, 2006

Performance evaluation of TCP flows

Florian Hisch
Advisor: Fabien Geyer
Innovative Internet Technologies and Mobile Communication SS 2014
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: hisch@in.tum.de

ABSTRACT
The behavior of different TCP algorithms has been subject to researchers over the past two decades. Various models have been developed to suit diverse network and traffic structures. This paper presents some of those analytic models and compares them with respect to their assumptions and validation techniques. It also classifies the given models and finally investigates their applicability on different network types.

Keywords
TCP modeling, Fluid model, IEEE802.11, AQM, congestion control

1. INTRODUCTION
Several analytic TCP (Transmission Control Protocol) models, like [3], [5], [7], [12], [14], [15], [16] and [17], have been proposed in the last decades. This amount of different models is justified by the fact that over 90% of network traffic is transported by TCP.
The models are used to show and improve some TCP characteristic like the throughput, responsiveness or TCP friendliness. Models are also used to estimate the overall performance of a network topology. [10]

The large amount of different TCP models necessitate a comparison and survey on the assumptions and validation techniques. This paper reviews the outcome made by Khalifa et al.[10]. It updates their results on the basis of more recent developments and extends them with models for wireless networks. The rest of this paper is structured as follows: Section 2 gives an overview about TCP and Active Queue Management (AQM). The models are classified in Section 3 and discussed (in order of assumptions and network restrictions) in Section 4. A comparison of their applicability in different network topologies is made in Section 5. Conclusions are given in Section 6.

2. OVERVIEW OF TCP AND AQM
The *Transmission Control Protocol* (TCP) is one of the most used transport protocols (ISO/OSI level 4). In contrast to UDP (User Datagram Protocol) it's called a 'connection-oriented'-protocol, because it establishes a bidirectional connection between two end-points.

The protocols on network level (ISO/OSI level 3), such as IP, are packet oriented. These packets can arrive in any order, duplicated or can be even dropped by any point in the network. Latter will be simply called a packet *loss*. These losses are neither reported to the packet source nor to its destination.
A TCP connection is used to ensure the data exchange and comes therefore with several features like Three-Way-Handshake, Sequence-Numbers and Acknowledgments. It also tries to address two major issues in networking: flow control and congestion control.

The data to be send is organized in a Sliding Window, that limits the amount of data which can be send at once. All bytes (octets) get a consecutive sequence number which is also used to identify the packets on network level. The source sends WND (window size) bytes into the network. Those bytes are combined to segments which are acknowledged by the receiver with its sequence-number.

Flow control makes sure that the source never sends a higher amount of data than the receiver can handle. A field in the TCP header shows the current space in the receiver-buffer RWND.
Congestion control assumes that the probability of an IP-packet loss because of a transmission error is, thanks to channel-coding, very low (approx. 10^{-9} with 8b10b coding and Cat-5e cable) [18]. Most losses occur due to network congestion and thus induced packet drops. The congestion control uses a Congestion Window to determine how many bytes the network could handle. The size of this window is labeled CWND. [4]

The size of the Sliding Window WND is the minimum of the Receiver Window RWND and Congestion Window CWND. All reviewed papers assume that the Receiver Window RWND is as big as possible. That said, the Sliding Window size is only related to the Congestion Window CWND.

Many RFCs, like [9], [19] or [2], suggest different algorithms to calculate CWND. They have the most impact on the throughput of TCP flows and are therefore described in the following.

doi: 10.2313/NET-2014-08-1_11

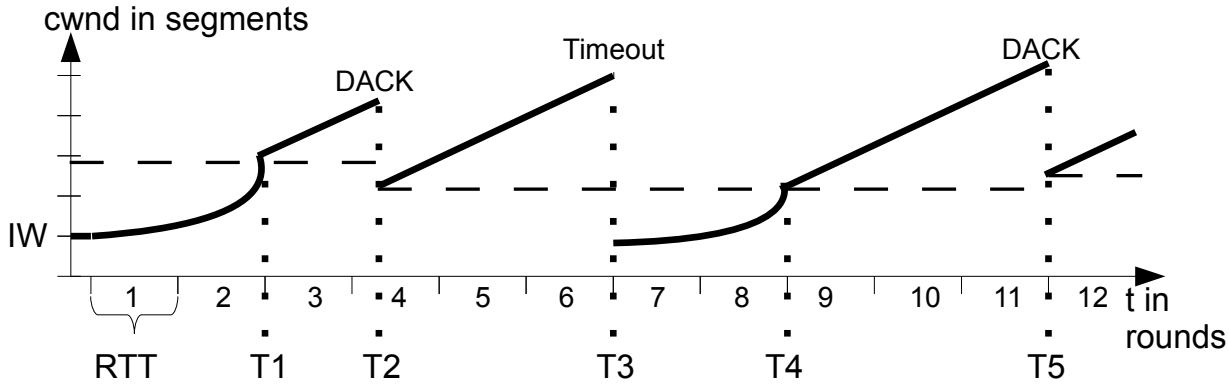


Figure 1: Example: TCP Congestion Window during Initial Slow Start and Congestion Avoidance (TCP Reno like). Horizontal dashed line is the current threshold ssthresh. IW marks the initial window and RTT the round-trip-time. DACK stands for: triple Duplicated ACK arrived.

TCP Slow Start

The Congestion Window CWND is initially set to a small default value, e.g. $\text{IW} = \min(4 \cdot \text{SMSS}, \max(2 \cdot \text{SMSS}, 4380))$ in bytes [9]. The SMSS is the size of the largest segment that the sender can transmit.

For each acknowledged packet, the Congestion Window increases by one SMSS. Thus, the window grows roughly exponential by $\text{CWND}(t+1) = \text{CWND}(t) + \text{SMSS} \cdot \text{CWND}(t)$ after CWND bytes were send. The exact growing policy is subject to miscellaneous TCP versions like TCP Reno or TCP Vegas .

Slow Start ends and Congestion Avoidance begins when CWND exceeds a given threshold SSTHRESH or when congestion is detected. Latter is observed due to a Timeout or Duplicated Acknowledgments DACKs.

Congestion Avoidance

During Congestion Avoidance the size of the Congestion Window increases only linear. TCP Reno for example increases with $\text{CWND} = \text{CWND} + \text{SMSS} \cdot \text{SMSS}/\text{CWND}$ for each received ACK. Doing so, the network load converges to its maximum in smaller steps to avoid a sudden overload.

SSTHRESH and CWND are reduced if congestion was detected. The exact reduction-behavior depends on the used TCP version. TCP Reno for example distinguishes between

3-Duplicated-ACKs:	$\text{SSTHRESH} = \text{CWND}/2$ $\text{CWND} = \text{CWND}/2$ start with Congestion Avoidance
Timeout :	$\text{SSTHRESH} = \text{CWND}/2$ $\text{CWND} = \text{IW}$ start with Slow Start

Figure 1 shows an example developing of the Congestion Window size CWND over the time. At time $t = 0$ the window size is given by the initial window IW. In the section $0 < t < T1$ CWND grows exponential until CWND > SSTHRESH ($t = T1$). The window increases only linear until three Duplicated ACKs (DACKs) are received ($t = T2$) when CWND is halved and SSTHRESH is adjusted accordingly. TCP is in Congestion Avoidance phase until $t = T3$ where a Timeout (TO) occurred. Again, CWND and SSTHRESH are adapted, but the Timeout has triggered a Slow Start . After reaching the threshold ($t = T4$), TCP is back in Congestion Avoidance again.

Fast Retransmit / Fast Recovery

A Duplicated ACK could be generated when the packets arrive out-of-order. This can be caused by full buffers, a re-ordering of network packets or a replication of the data packet or the ACK itself. RFC 5681 suggest to use a 'fast retransmit' algorithm to detect Duplicated ACKs and to perform an immediate retransmission of the lost data.

After this retransmission, the 'fast recovery' algorithm sends new data ignoring Duplicated ACKs until a non-duplicate ACK is received. [2] recommends, not to perform Slow Start but to artificially inflate the Congestion Window. This is legitimate because the Duplicated ACKs state that there were already packets received, which don't participate in the network traffic anymore. If ACKs for the previously missed packets arrive, the Congestion Window is deflated to its original size before Fast Retransmit was done.

Active Queue Management

Buffers are used by routers to accommodate stress peaks in traffic and to keep a reserve, so that the link doesn't go idle [20]. Has the buffer reached his maximum amount of packets, further received packets have to be dropped. The method of simply fill and drop the rest is known as Taildrop algorithm.

Floyd et al. [8] show that the behavior of Taildrop leads to an oscillation between nearly idle and highly congested due to TCP flow synchronization. Active Queue Management (AQM) algorithms tries to avoid this synchronization

doi: 10.2313/NET-2014-08-1_11

by a more 'intelligent' dropping policy. RED (Random Early Detection) calculates for every incoming packet a drop probability. Doing so, a packet can be randomly dropped even if the buffer would have enough space to handle it. Enhancements of RED consider QoS (Quality of Service) factors to calculate the drop probability which prefer real-time flows like VoIP.

3. TCP MODEL CLASSIFICATION

Khalifa et al. [10] introduce three different classes for TCP models. The first one characterizes a TCP flow by its average throughput. The second class pays attention to the interaction between TCP flows, whereas the last class deals with TCP dynamics.

This paper presents a new classification, which distinguishes between Single connection models (SCM), Multiple connections models (MCM) and Fluid models.

3.1 Single connection models

This class observes a single TCP flow and its throughput in a single bottleneck network. Padhye et al. [15] give a model for the steady-state TCP Reno throughput of bulk transfers. This model was used by Parvez et al. [16] to develop a model for the more up-to-date TCP version NewReno. Cardwell et al. [5] introduce in contrast to the above ones, a model for a short-lived TCP flow. The papers [16] and [5] are reviewed more in detail below.

Three-Way-Handshake

The Three-Way-Handshake is used to establish the connection between two TCP endpoints. [5] et al. consider two phases:

- The initiating host sends $i \geq 0$ SYN packets until the $(i+1)$-th SYN arrives successfully. The probability of a successful packet transfer is p_f (forward probability).
- The receiver of the SYN packets answers with $j \geq 0$ SYN/ACK packets until the $(j+1)$-th SYN/ACK arrives successfully. The probability of a successful packet transfer is p_r (reverse probability).

Let $P_h(i,j)$ the probability of having a successful Three-Way-Handshake with exact i respectively j packet losses (see above).

$$P_h(i,j) = p_r^i \cdot (1-p_r) \cdot p_f^j \cdot (1-p_f)$$

The probability that the latency D_h, of a general Three-Way-Handshake, is t seconds or less, is

$$P[D_h \leq t] = \sum_{D_h(i,j) \leq t} P_h(i,j)$$

where $D_h(i,j)$ is the latency for exact i and j losses. $D_h(i,j)$ is given as

$$D_h(i,j) = \mathrm{RTT} + (2^i + 2^j - 2) \cdot D_s$$

RTT describes the average round trip time between the two points of the TCP flow. D_s is the minimum waiting time for a answer during Three-Way-Handshake. This time doubles with every try.

Cardwell et al. [5] present an approximation of the expected handshake time for loss rates low enough that most Three-Way-Handshakes succeed before TCP gives up.

$$E[D_h] = \mathrm{RTT} + D_s \cdot \left(\frac{1-p_r}{1-2p_r} + \frac{1-p_f}{1-2p_f} - 2 \right) \quad (1)$$

Initial Slow Start

After a successful Three-Way-Handshake, TCP starts with an Initial Slow Start . Cardwell et al. [5] observe two cases: The first one lets the Congestion Window increase unlimited, whereas the second case gives a maximum window size $W_{\max}$. To compare this model to the other ones, $W_{\max}$ will be set to infinity.

Let $E[S_{\mathrm{ss}}]$, the expected number of data segments send before a loss occurs, be

$$\begin{aligned} E[S_{\mathrm{ss}}] &= \left(\sum_{k=0}^{S-1} (1-p)^k \cdot p \cdot k \right) + (1-p)^S \cdot S \\ &= \frac{(1-(1-p)^S(1-p)}{p} + 1 \end{aligned}$$

where S is the total amount of data to be send in the whole TCP flow and p the probability for a segment loss. p does not distinguish between forward and reverse probability anymore.

CWND increases during Slow Start with

$$\begin{aligned} \mathrm{CWND}_{i+1} &= \mathrm{CWND}_i + \mathrm{CWND}_i / b \\ &= \gamma \cdot \mathrm{CWND}_i \end{aligned}$$

with $\gamma = (1 + 1/b)$ and w_1 as the initial window size.

The time needed to send all segments in a Congestion Window is further referred as a 'round'.

$$i = \log_\gamma \left(\frac{S_i(\gamma - 1)}{w_1} + 1 \right)$$

is the count of rounds needed to send S_i segments.

The formulas for i and $E[S_{\mathrm{ss}}]$ lead to the expected time to send $E[S_{\mathrm{ss}}]$ data segments during Slow Start:

$$E[D_{\mathrm{ss}}] = \mathrm{RTT} \cdot \log_\gamma \left(\frac{E[S_{\mathrm{ss}}](\gamma - 1)}{w_1} + 1 \right) \quad (2)$$

Congestion Avoidance / Fast Recovery

TCP distinguishes two ways to detect a packet loss: Triple-Duplicate-ACKs (DACK) and Timeouts (TO). Parvez et al. [16] derive a model without Timeouts (NoTO) first and extends it with the missing behavior later.

The evolution of the Congestion Window during Congestion Avoidance and Fast Recovery can be modeled as statistically

doi: 10.2313/NET-2014-08-1_11

identical cycles. Those cycles consists of several rounds, like the ones described above. Each cycle (CAFR) combines a Congestion Avoidance period and a Fast Recovery period. The throughput of one of those cycles is given as

$$E[T_{\text{NoTO}}] = \frac{E[S_{\text{CAFR}}]}{E[D_{\text{CAFR}}]}$$

where $E[S_{\text{CAFR}}]$ is the expected number of successfully transmitted segments and $E[D_{\text{CAFR}}]$ the duration for this period.

Parvez et al. [16] use, besides the packet loss probability p, also a segment loss rate q within a loss event. This indicates how many segments relative to all segments in a round are affected by a packet loss. During the Three-Way-Handshake and the initial Slow Start, q was set to 1, meaning that if a packet loss occurs, all segments in that round are lost.

The expected amount $E[m]$ of uniformly spaced drops in a window of size W can be obtained as

$$\begin{aligned} E[m] &= \sum_{m=1}^{W-3} m \cdot f(m-1, W-1, q) \\ &= \sum_{m=1}^{W-3} m \cdot \binom{W}{m} \cdot q^{m-1} \cdot (1-q)^{W-m} \\ &\approx 1 + (W-1) \cdot q \approx 1 + W \cdot q \end{aligned}$$

The upper border for the sum $W-3$ is valid, as there has to be a DACK. $f(m-1, W-1, q)$ is obviously the Binomial probability mass function.

$E[S_{\text{CAFR}}]$ and $E[D_{\text{CAFR}}]$ can be split in a Linear Increase (LI), β and Fast Recovery (FR) phase. For example $E[S_{\text{CAFR}}] = E[S_{\text{LI}}] + E[S_\beta] + E[S_{\text{FR}}]$.
The most interesting value is $E[S_{\text{FR}}]$, because of the inflation respectively the deflation of the Congestion Window . Each DACK increases CWND by one segment. This behavior leads to the possibility that during Fast Recovery not only missing (lost) data is transmitted, but also new data. If $m > \frac{W}{2}$, TCP will not transmit any new data. Parvez et al. [16] obtain

$$\begin{aligned} E[S_{\text{FR}}]^{m \le \frac{W}{2}} &= \sum_{j=1}^{m} \left(\frac{W}{2} - m + j - 1 \right) = \frac{m}{2} \cdot (W - m - 1) \\ E[S_{\text{FR}}]^{m > \frac{W}{2}} &= \sum_{k=m-\frac{W}{2}+1}^{m-1} \left(\frac{W}{2} - m + k \right) = \frac{W^2}{8} - \frac{W}{4} \\ E[S_{\text{FR}}] &= \sum_{m=1}^{\frac{W}{2}} f(m-1, W-1, q) \cdot S_{\text{FR}}^{m \le \frac{W}{2}} \\ &+ \sum_{m=\frac{W}{2}+1}^{W-3} f(m-1, W-1, q) \cdot S_{\text{FR}}^{m > \frac{W}{2}} \end{aligned}$$

$$\approx \frac{W^2}{2}(q - q^2) + \frac{W}{2}(1 - 5q + 3q^2) - (1 - 2q + q^2)$$

where $f(m-1, W-1, q)$ is again the Binomial probability mass function.

The remaining expressions of those phases are:

$$\begin{aligned} E[S_{\text{LI}}] &= \sum_{i=\frac{W}{2}}^{W} i = \frac{3}{8}W^2 + \frac{3}{4}W \\ E[S_\beta] &= \frac{\text{W}}{2} \\ E[D_{\text{LI}}] &= \left(\frac{W}{2} + 1 \right) \cdot \text{RTT} \\ E[D_\beta] &= \frac{\text{RTT}}{2} \\ E[D_{\text{FR}}] &= E[m] \cdot \text{RTT} \approx (1 + Wq) \cdot \text{RTT} \end{aligned}$$

With the expressions above the total terms $E[S_{\text{CAFR}}]$ and $E[D_{\text{CAFR}}]$ are:

$$E[S_{\text{CAFR}}] =$$

$$\left(\frac{3}{8} + \frac{q}{2} - \frac{q^2}{2} \right) W^2 + \left(\frac{7}{4} - \frac{5q}{2} + \frac{3q^2}{2} \right) W - \left(1 - 2q + q^2\right) \quad (3)$$

$$E[D_{\text{CAFR}}] =$$

$$\left(\frac{W}{2} + Wq + \frac{5}{2} \right) \text{RTT} \quad (4)$$

The formulas above need an approximation for the Congestion Window size W. The size can be expressed in terms of p and q as

$$W \approx \frac{10pq - 5p + \sqrt{p(24 + 32q + 49p}}{p(3 + 4q)} \quad (5)$$

To extend the NoTO model and to include also the Timeout case, Parvez et al. [16] adapt their throughput expression given in (6). p_{TO} is the probability that a loss leads to a Timeout. $E[D_{\text{CA}}] + E[D_{\text{TO}}] + E[D_{\text{SS}}]$ combines a Congestion Avoidance (CA) phase with the Timeout itself and the consecutive Slow Start (SS) phase. From the NoTO model follows that $E[D_{\text{CA}}] = E[D_{\text{LI}}] + E[D_\beta]$ and $E[S_{\text{CA}}] = E[S_{\text{LI}}] + E[S_\beta]$. Since TCP forgets all outstanding packets after a Timeout, $E[S_\beta] = E[D_\beta] = 0$ and thus $E[S_{\text{CA}}] = E[S_{\text{LI}}]$ respectively $E[D_{\text{CA}}] = E[D_{\text{LI}}]$.

The derivation of p_{TO} has to be split up in $p_{\text{TO}} = p_{\text{DTO}} + p_{\text{IFR}}$, a direct Timeout during Congestion Avoidance and a Timeout during Fast Recovery. First one is detected by more than $W-3$ segments are lost. From the Binomial probability mass function f follows

$$p_{\text{DTO}} = \sum_{m=W-2}^{W} f(m-1, W-1, q)$$

The latter case probability derivation is rather complex and includes several assumptions (see section 4). Therefore only the result is presented

$$p_{\text{IFR}} = \sum_{m=1}^{W-3} f(m-1, W-1, q) \left[1 - (1-p)^{\frac{mW}{2}} \right]$$

doi: 10.2313/NET-2014-08-1_11

$$E[T_{\text{Full}}] = \frac{(1-p_{\text{TO}})E[S_{\text{CAFR}}] + p_{\text{TO}}(E[S_{\text{CA}}] + E[S_{\text{TO}}] + E[S_{\text{SS}}])}{(1-p_{\text{TO}})E[D_{\text{CAFR}}] + p_{\text{TO}}(E[D_{\text{CA}}] + E[D_{\text{TO}}] + E[D_{\text{SS}}])} \tag{6}$$

The total probability for a Timeout is given as

$$p_{\text{TO}} = 1 - \sum_{m=1}^{W-3} f(m-1, W-1, q)\left[(1-p)^{\frac{mW}{2}}\right] \tag{7}$$

Obviously TCP transmits no new data during a Timeout, thus $E[S_{\text{TO}}] = 0$. T_0 is the initial Timeout duration which doubles with each try. So, the Timeout of the i-th retransmission is $T_i = 2^{i-1}T_0$. The probability that an i-th retransmission is needed, is $\frac{p^{i-1}}{1-p}$ (geometric series). The total time spend during the Timeout is

$$E[D_{\text{TO}}] = T_0 \frac{1 + p + 2p^2 + 4p^3 + 8p^4 + 16p^5 + 32p^6}{1-p}$$

In the following Slow Start phase, the Congestion Window starts with size 1 and doubles each round until it reaches $\frac{W}{2}$. Similar to the derivation of the Initial Slow Start phase follows

$$E[S_{\text{SS}}] = 1 + 2 + 4 + \ldots + \frac{W}{4} = 2^{1+\log\frac{W}{4}} - 1$$

$$E[D_{\text{SS}}] = \left(\log\frac{W}{4} + 1\right)\text{RTT}$$

With the expressions above the total terms S_{Full} and D_{Full} are:

$$E[S_{\text{Full}}] = \frac{1}{p} + \frac{W^2 q}{1 + Wq} \tag{8}$$

$$E[D_{\text{Full}}] = E[N]\cdot\text{RTT} + p_{\text{TO}}\left((1 + 2p + 4p^2)T_0 + \left(1 + \log\frac{W}{4}\right)\text{RTT}\right) \tag{9}$$

where $E[N] = \left(\frac{W}{2} + \frac{3}{2} + (1-p_{\text{TO}})(1+Wq)\right)$.

Model for Short-Lived Transfers

Many TCP flows in today's Internet, transport only a small amount (less than 10 KB) of data and will therefore never leave the Slow Start phase. Cardwell et al. [5] used this statistical information to derive a model which focuses on the Three-Way-Handshake and Slow Start.
The throughput of a Short-Lived TCP transfer is approximated by:

$$E[T_{\text{short}}] = \frac{S}{E[D_{\text{ss}}] + \frac{(S - E[S_{\text{ss}}])}{E[T_{\text{Full}}]}} \tag{10}$$

The exact expression given by Cardwell et al. [5] also consider the delays $E[D_{\text{loss}}]$ and $E[D_{\text{delack}}]$ which are caused by the first loss and delayed ACKs (Nagle algorithm).

3.2 Multiple connections models

This class observes many TCP flows and their throughput in a multiple bottleneck network. Firoiu et al. [7] are using their results in the single bottleneck case to derive also a model for the steady-state TCP throughput in larger networks. In contrast to Single connection models is a detailed analysis of startup effects (e.g. Three-Way-Handshake) due to the parallelism of the many TCP flows not possible. Though, heterogeneous traffic, like TCP and UDP, and Active Queuing Management (e.g. RED) can be modeled additionally.

Because of the similarity of Multiple connections models to Single connection models, a detailed derivation of the throughput is skipped. Nevertheless, the throughput at the k-th link in the network is given by

$$T_k\left(\left\{\{\overline{q}_m\}_{i,j}\right\}_k\right) = \sum_i^{\{f_i\}_k} t_i\left(\{\overline{q}_k\}\right) + \sum_j^{\{u_j\}_k} r_j\left(p'\right) \tag{11}$$

with the average queue length on link k $\overline{q}_k$ and p' the overall drop rate over multiple bottlenecks. $t_i(.)$ and $r_j(.)$ is the throughput of the i-th TCP flow and j-th UDP flow. The total throughput of the whole network is the sum over all $T_k\left(\left\{\{\overline{q}_m\}_{i,j}\right\}_k\right)$.

3.3 Fluid models

Fluid models use adjacency matrices to define a network. It distinguishes between Sources and Links, which both have a different view on the traffic. This can be used to model not only TCP's congestion control but also some Active Queuing Management (AQM).

Network model

A typical approach is to define a set $\mathbb{L}$ of links with finite capacities $c = (c_l, l \in \mathbb{L})$. Sources are centralized in the set $\mathbb{S}$ and every of these sources can use $\mathbb{L}_s \subseteq \mathbb{L}$ of links. To connect the links with the sources a routing matrix

$$R_{ls} = \begin{cases} 1, & \text{if } l \in \mathbb{L}_s \\ 0, & \textit{otherwise} \end{cases} \tag{12}$$

is defined. Each source is associated with its transmission rate $x_s(t)$, whereas the links are associated with a congestion measure $p_l(t) \geq 0$ at time t. Using the routing matrix R_{ls} the following expressions can be derived:

$$\begin{aligned} y_l &= \textstyle\sum_s R_{ls} x_s(t) && \text{aggregated source rate at link } l \\ q_s &= \textstyle\sum_l R_{ls} p_l(t) && \text{end-to-end congestion for sources} \\ y(t) &= Rx(t) \\ q(t) &= R^T p(t) \end{aligned}$$

doi: 10.2313/NET-2014-08-1_11

Like in reality, a source s can only observe its own traffic rate $x_s(t)$ and the end-to-end congestion $q_s(t)$. Link l can only observe its own congestion $p_l(t)$ and the flow rate $y_l(t)$.

To model the adjustments on transmission rate and congestion measure Low [14] uses three state functions F, G, H. They are defined as

$$x_s(t+1) = F_s(x_s(t), q_s(t)) \quad (13)$$
$$p_l(t+1) = G_l(y_l(t), p_l(t), v_l(t)) \quad (14)$$
$$v_l(t+1) = H_l(y_l(t), p_l(t), v_l(t)) \quad (15)$$

The internal state vector $v_l(t)$ gives the opportunity to store information, such as queue lengths. A specific set of F, G, H functions is the essential part of a Fluid model. It contains the whole behavior of TCP (F) and AQM (G, H).

To get a steady-state, the F, G, H functions above shall reach an equilibrium point (x, p) so that $x_s = F_s(x_s, q_s)$. To get the end-to-end congestion in the specific equilibrium point, an additional function f_s is defined with $q_s = f_s(x_s) > 0$. This can be used to identify the equilibrium point without congestion $(x_s, 0)$. Obviously $F_s(x_s, 0) = x_s$ and $f_s(x_s) = 0$. The existence of these equilibrium points is subject to several assumption respectively conditions and not all Fluid models can comply with all [17].

To determine the overall performance of the network in a steady-state equilibrium point, a utility function for each source s is defined as

$$U_s(x_s) = \int f_s(x_s)dx, \qquad x_s \geq 0$$

This utility function can be used to maximize the aggregated utility given by an equilibrium point x_s by

$$\max_{x \geq 0} \sum_s U_s(x_s), \qquad \text{subject to } Rx \leq c \quad (16)$$

under the premise that the flow rate $y = Rx(t)$ never exceeds the capacities c.

(F,G,H)-models

Some definitions that are used in the following are adjusted from the original papers to suit the definitions made in previous models:
Let W_s be the Congestion Window size of source s and let RTT_s be the equilibrium round-trip time which is assumed to be constant. The average source rate $x_s(t)$ is defined by $x_s(t) = W_s(t)/\mathrm{RTT}_s$, since the time is discrete and is on the order of several round-trip times.

Exemplary derivation of a F model for TCP Reno:
Since Fluid models can only observe the steady-state of the network, Three-Way-Handshake and Slow Start are no subject. One variant of TCP Reno halves the Congestion Window every time congestion is detected. This congestion detection is done by 'marks'. Let $p_l(t)$ be the current marking probability at link l, then the end-to-end loss probability is

$$q_s(t) = \sum_l R_{ls} p_l(t)$$

This definition is only valid if $0 \leq q_s(t) \leq 1$, so the probabilities p_l have to be small enough. On average, source s receives $x(t)(1 - q_s(t))$ positive acknowledgments per unit time. Each positive ACK increases W_s by $1/W_s(t)$ (Congestion Avoidance). The $x(t)q_s(t)$ negative ACKs (marks) per unit time halves the Congestion Window. In total, the Congestion Window changes on average by

$$x_s(t)(1 - q_s(t)) \cdot \frac{1}{W_s(t)} - x_s(t)q_s(t) \cdot \frac{1}{2} \cdot \frac{4W_s(t)}{3}$$

The TCP Reno algorithm in $F_s(x_s(t), q_s(t))$ shape is given as

$$x_s(t+1) = F_s(x_s(t), q_s(t))$$
$$= \left[x_s(t) + \frac{1 - q_s(t)}{\mathrm{RTT}_s^2} - \frac{2}{3} q_s(t) x_s^2(t)\right]^+ \quad (17)$$

Using the F function above, the $f_s(x_s) = q_s$ function for TCP Reno is

$$q_s = \frac{3}{2x_s^2 \mathrm{RTT}_s^2 + 3} =: f_s(x_s)$$

and the utility function $U_s(x_s)$ can be expressed as

$$U_s(x_s) = \frac{\sqrt{3/2}}{\mathrm{RTT}_s} \tan^{-1}\left(\sqrt{\frac{2}{3}} x_s \mathrm{RTT}_s\right) \quad (18)$$

With only one internal state in the functions G, H, it is not possible to model abrupt changes in the queuing behavior. For example, there is only an approximation for Drop-Tail since the dropping immediately starts if the buffer is full. Dropping algorithms like RED (Random Early Drop) and REM (Random Exponential Marking) which are more 'smooth', can be modeled without any discrepancy. Exemplarily derivation of a G, H model for RED:

RED consists of two internal variables, the current queue length $b_l(t)$ and the average queue length $r_l(t)$, which is calculated with a factor $\alpha \in (0, 1)$:

$$b_l(t+1) = [b_l(t) + y_l(t) - c_l]^+$$
$$r_l(t+1) = (1 - \alpha_l) r_l(t) + \alpha_l b_l(t)$$

The marking (loss) probability of the link l is given as a function of the average queue length $r_l(t)$. There are three additional RED characteristics: $\underline{b_l}$ and $\overline{b_l}$ being a lower / upper threshold and $m_l \in (0, 1)$ being a value to adjust the stringency. If m_l is nearly 0, RED will start dropping some packets if the queue is longer than $\overline{b_l}$. If m_l is nearly 1, RED will drop every packet if the queue length exceeds $\overline{b_l}$. The exact expression for the RED dropping behavior is:

$$p_l(t) = \begin{cases} 0, & r_l(t) \leq \underline{b_l} \\ \frac{m_l}{\overline{b_l} - \underline{b_l}}(r_l(t) - \underline{b_l}), & \underline{b_l} \leq r_l(t) \leq \overline{b_l} \\ \frac{1 - m_l}{\overline{b_l}}(r_l(t) - \overline{b_l}), & \overline{b_l} \leq r_l(t) \leq 2\overline{b_l} \\ 1, & otherwise \end{cases}$$

doi: 10.2313/NET-2014-08-1_11

To write the formulas above in the G, H form, the internal state vector hast to be:

$$p_l(t+1) = G_l(y_l(t), p_l(t), v_l(t)) \tag{19}$$

$$v(t+1) = \begin{pmatrix} b_l(t+1) \\ r_l(t+1) \end{pmatrix} = H_l(y_l(t), p_l(t), v_l(t))$$

$$= \begin{pmatrix} [b_l(t) + y_l(t) - c_l]^+ \\ (1-\alpha_l)r_l(t) + \alpha_l b_l(t) \end{pmatrix} \tag{20}$$

4. ASSUMPTIONS AND VALIDATION

Before making any assumptions, there is not a generally accepted TCP implementation. The RFCs describing the behavior of TCP are full of recommendations and thus TCP implementations vary very much. Therefore, assumptions have to be made to define the TCP environment. These assumptions can be categorized in classes regarding the Data Transfer, End Points and Network.

Data Transfer

The availability of the data to be sent is one of the first assumptions made during the modeling process. Usually, it is assumed that the application layer generates the data without any latency. This assumption is only true if the application can generate data faster than the TCP connection can send it. Compared to nowadays communications technologies (Ethernet, Wifi, ...), computer memory and CPUs are rarely the bottleneck in TCP transfers. Tough, for special cases, like the direct point-to-point high speed connection (e.g. via Gigabit LAN), the transmit queue may run empty.

Most of the paper propose a predefined amount of data or rather the Data transfer length d. Knowing this length, it is possible to estimate how much the initial TCP behavior (Three-Way-Handshake, Initial Slow Start) affects the over all throughput. Is d rather small (around 10 KB), Short-Lived models like [5], which consider the initial TCP behavior, are more accurate than the Steady-State models. Those ignore the start-up effects and simplify their model by considering only Congestion Avoidance and Fast Recovery algorithms.

Fluid models are Steady-State models with an unlimited amount of data. They can give no statement, how much data is needed to reach the equilibrium point of the network. To get an approximation for the needed data length or the time to reach this point, expressions from the Single connection models can be used.

End Points

TCP implementations and TCP versions vary, as already mentioned, very much. This affects especially the congestion control. Three different TCP versions are referred in the papers: TCP Tahoe (1988), TCP Reno (1990) respectively TCP NewReno (1999) and TCP Vegas (1995-2005). TCP Reno is the version, which is implemented in the most operating systems. [1]

Most papers give their model in different variations for the loss detection algorithms of some of the TCP versions. Additional assumptions regarding the End Points are

- the average number of segments acknowledged by a single ACK,
- the size of one segment,
- the size of the window limitation (the size of the Receiver Window), which can be also set to infinity, and
- timing constraints (duration of initial Timeout, Three-Way-Handshake Timeout , ...).

A common assumption is to neglect additional algorithms like the Nagle algorithm and silly window syndrome.

Network

Since TCP is theoretical independent from the network layer, it can be used on any network topology. Nevertheless TCP throughput is always a function of the loss probability p which depends on the used network. Hence, models never specify the exact topology, but are more accurate in a certain network category. These categories can be distinguished by means of tethered / wireless connections and whether there are single or multiple bottlenecks.

A typical assumption (found in all reviewed models) is to neglect transmission and processing delays. Further, the round-trip time RTT is assumed to be independent from CWND and in most cases even constant. Doing so, the model becomes less complex, because a variable RTT would add an additional stochastic process. This assumption is however not justifiable, since nobody knows how the TCP packets are routed through the network, fragmented or delayed by Active Queuing Management. Even TCP itself has to measure the RTT during the whole connection time to adjust the Retransmission Timeout duration RTO to the current network situation. Hence, RTT cannot be assumed to be constant.
Wired networks, especially those with only direct links (no switches), have a small RTT with a little variance. Most network layers for wireless networks (e.g. 802.11x, Wireless LAN) use CSMA/CA, which contains a random waiting time after a collision has occurred. Thus, wireless connections have a bigger RTT with a higher variance.

Table 3 summarizes some of the characteristics of all reviewed models.

Model Validation

All models have to be proofed, how accurately they reproduce the reality. They can be validated by comparing the calculated throughput with results from: simulations, controlled measurements under conditions that suit the assumptions and live measurements from the Internet. Table 1 shows some of the advantages and disadvantages of the said validation methods. Table 2 shows the validation methods used by the miscellaneous papers.

doi: 10.2313/NET-2014-08-1_11

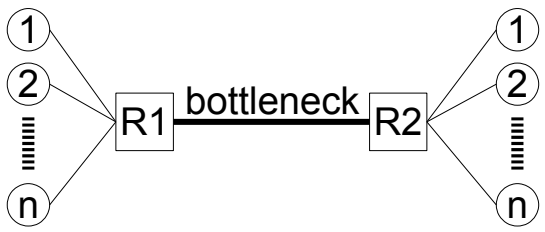


Figure 2: Single bottleneck layout. n sources and sinks are connected via a single link which has more incoming data than it can transmit.

5. APPLICABILITY

None of the models is applicable to every network topology. The applicability can be defined on the basis of some assumptions. Table 4 shows the network restrictions that can be derived from the assumptions. For example, if the RTT is expected to be constant, the network topology has to be tethered or contains not more than one single wireless hop. This relation can be explained by the used physical layers (mostly Ethernet respectively 802.11x) and especially by the different CSMA variants (Collision Detection / Collision Avoidance).

Most models consider a single bottleneck. This is however given only in small networks. Firoiu et al. [7] define that a link is a bottleneck when the arrival rate of incoming packets is higher than service rate (departure rate), and the packets are enqueued. Thus, a network topology has to be like figure 5 to be called single bottleneck. Typical networks are more complex and hence contain multiple bottlenecks. The transition of a single to a multiple bottleneck model and their resulting (relative) error is shown in [7]. Fluid models consider multiple bottlenecks per default as there are all link capacities known.

6. CONCLUSION

This paper presents an extended overview of TCP models based on the previous summary made by Khalifa et al. [10]. It classified the models in Single connection models, Multiple connections models and Fluid models, compared them and surveyed their validation and applicability.

Single connection models (SCM) can reproduce TCP in a microscopic way (considering Three-Way-Handshake, Slow Start, Congestion Avoidance and Fast Recovery) but loose accuracy in complex networks with multiple bottlenecks. Multiple connections models (MCM) extend SCM and are used in complex networks with multiple bottlenecks. They reproduce the steady-state throughput of many TCP flows and can model heterogeneous traffic (e.g. TCP and UDP flows). In contrast, Fluid models observe only the macroscopic behavior of TCP but is accurate also in complex networks. Fluid models can be easily extended to consider Active Queuing Management.

The comparison showed that most of the models are only validated by simulation. The complex behavior of TCP, with its not standardized implementations, can generate side-effects which lower the simulated throughput. Only two models have been validated by live measurements in the Internet. Future models should be validated especially by live measurements and comparisons to existing models, to show their achievements.

The current extensive growth of wireless networks require new models for the TCP throughput. Most of the reviewed models assume characteristics of a tethered network which makes them inaccurate in wireless (especially Ad-Hoc) networks. Future models have to consider both network types.

7. REFERENCES

[1] A. Afanasyev, N. Tilley, P. Reiher, and L. Kleinrock. Host-to-host congestion control for TCP. *IEEE Communication Surveys and Tutorials*, 12(3):304–342, July 2010. (2012 The IEEE Communications Society Best Tutorial Paper Award).

[2] M. Allman, V. Paxson, and E. Blanton. TCP Congestion Control. RFC 5681 (Draft Standard), Sept. 2009.

[3] E. Altman, K. Avrachenkov, and C. Barakat. A Stochastic Model of TCP/IP with Stationary Random Losses. *SIGCOMM Comput. Commun. Rev.*, 30(4):231–242, Aug. 2000.

[4] A. Badach and E. Hoffmann. *Technik der IP-Netze - TCP/IP incl. IPv6 - Funktionsweise, Protokolle und Dienste, 2. erw. Aufl.* Carl Hanser Verlag, München, 2007.

[5] N. Cardwell, S. Savage, and T. Anderson. Modeling TCP latency. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1742–1751 vol.3, Mar 2000.

[6] V. Firoiu, J.-Y. Le Boudec, D. Towsley, and Z.-L. Zhang. Theories and models for Internet quality of service. *Proceedings of the IEEE*, 90(9):1565–1591, Sep 2002.

[7] V. Firoiu, I. Yeom, and X. Zhang. A framework for practical performance evaluation and traffic engineering in IP networks. *IEEE ICT 2001*, 2001.

[8] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *Networking, IEEE/ACM Transactions on*, 1(4):397–413, Aug 1993.

[9] V. Jacobson, R. Braden, and D. Borman. TCP Extensions for High Performance. RFC 1323 (Proposed Standard), May 1992.

[10] I. Khalifa and L. Trajkovic. An overview and comparison of analytical TCP models. In *Circuits and Systems, 2004. ISCAS '04. Proceedings of the 2004 International Symposium on*, volume 5, pages V–469–V–472 Vol.5, May 2004.

[11] X. Lan, S. Li, and S. Zhang. Network Coded TCP throughput Analysis in Wireless Mesh Networks. In *Mobile Ad-hoc and Sensor Networks (MSN), 2013 IEEE Ninth International Conference on*, pages 225–232, Dec 2013.

[12] X. Li, P.-Y. Kong, and K.-C. Chua. TCP Performance

doi: 10.2313/NET-2014-08-1_11

in IEEE 802.11-Based Ad Hoc Networks with Multiple Wireless Lossy Links. *Mobile Computing, IEEE Transactions on*, 6(12):1329–1342, Dec 2007.

[13] Y. Liu, F. Lo Presti, V. Misra, D. Towsley, and Y. Gu. Fluid Models and Solutions for Large-scale IP Networks. *SIGMETRICS Perform. Eval. Rev.*, 31(1):91–101, June 2003.

[14] S. Low. A duality model of TCP and queue management algorithms. *Networking, IEEE/ACM Transactions on*, 11(4):525–536, Aug 2003.

[15] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose. Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation. *IEEE/ACM Trans. Netw.*, 8(2):133–145, Apr. 2000.

[16] N. Parvez, A. Mahanti, and C. Williamson. An Analytic Throughput Model for TCP NewReno. *IEEE/ACM Trans. Netw.*, 18(2):448–461, Apr. 2010.

[17] Q. Peng, A. Walid, and S. H. Low. Multipath TCP Algorithms: Theory and Design. *SIGMETRICS Perform. Eval. Rev.*, 41(1):305–316, June 2013.

[18] E. Stein. *Taschenbuch Rechnernetze und Internet. Mit 94 Tabellen*. Fachbuchverl. Leipzig im Carl-Hanser-Verl., München, 3., neu bearb. aufl. edition, 2008.

[19] W. Stevens. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. RFC 2001 (Proposed Standard), Jan. 1997. Obsoleted by RFC 2581.

[20] D. Wischik and N. McKeown. Part I: Buffer Sizes for Core Routers. *SIGCOMM Comput. Commun. Rev.*, 35(3):75–78, July 2005.

doi: 10.2313/NET-2014-08-1_11

Table 1: Advanteges and disadvanteges of model validiation methods

	advantages	disadvantages
simulation	full knowledge on the environment all characteristics observable (e.g. queue lengths on all routers) measurement recording does not falsify the results	not simulated side-effects can affect throughput in reality simulation and model fails can compensate each other TCP implementation can differ from real implementation in operation systems
controlled measurements	partial knowledge on the environment, but enough to ensure the assumptions real-world side-effects affect the results	high effort to control the network environment measurement recordings partially falsify the result, because of additional traffic
live measurements	model can be validated against the reality	small knowledge of the environment (topology is rather complex)

Table 2: Model validation techniques.

Model	Simulations	Controlled meas.	Live meas.	Comparison
Padhye [15]	Yes	Yes	No	None
Cardwell [5]	Yes	Yes	Yes	[15]
Firoiu [7]	Yes	No	No	None
Low [14]	Yes	No	No	None
Liu [13]	Yes	No	No	None
Altman [3]	No	Yes	No	[15]
Li [12]	Yes	No	No	None
Parvez [16]	Yes	Yes	Yes	[15]
Peng [17]	Yes	No	No	[14]

Table 3: Summary of the models. The TCP algorithms are Three-Way-Handshake (TWH), Initial Slow Start (ISS), Congestion Avoidance (CA) and Fast Recovery (FRC). Losses are detected by Triple Duplicated ACKs (DACK) or Timeouts (TO). CA[1] and DACK[1] represent that Fluid models observe only the macroscopic behavior of TCP. [2]: Firoiu et al. introduce a SCM and a MCM, which means that TWH and ISS are only modled in the SCM.

Model	Type	Based on	Lengths	TCP algorithm	Loss detection	Data/ACK Losses	AQM
Padhye [15]	SCM		Long-lived	CA	DACK, TO	Bursty/None	Taildrop
Cardwell [5]	SCM	[15]	Short/Arbitrary	TWH, ISS, CA	DACK, TO	Bursty/SYN only	Taildrop
Firoiu [7]	MCM[2]	[15], [5]	Arbitrary	TWH, ISS, CA	DACK, TO	Bursty/SYN only	RED
Low [14]	Fluid		Long-lived	CA[1]	DACK[1]	/None	RED, REM
Liu [13]	Fluid		Long-lived	CA[1]	DACK[1]	Bursty/None	RED
Altman [3]	SCM	[15]	Long-lived	CA	DACK, TO	Stationary/None	Taildrop
Li [12]	SCM		Long-lived	CA, FRC	DACK, TO	Bursty/None	Taildrop
Parvez [16]	SCM	[15]	Long-lived	CA, FRC	DACK, TO	Adjustable/None	Taildrop
Peng [17]	Fluid	[14]	Long-lived	CA[1]	DACK[1]	Bursty/None	RED

Table 4: Assumptions and the validity on different network types. A X marks wether the RTT vaiance or the Loss probability variance is small or large.

		RTT		Losses	
		small	large	small	large
single bottleneck	tethered	X		X	
	wireless		X		X
multiple bottleneck	tethered	X			X
	wireless		X		X

doi: 10.2313/NET-2014-08-1_11

Formal Methods: Header Space Analysis

Benedikt Jaeger
Betreuer: Cornelius Diekmann
Hauptseminar - Innovative Internettechnologien und Mobilkommunikation SS2014
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: bene.jaeger@tum.de

KURZFASSUNG

Computernetzwerke sind heutzutage sehr groß und komplex, was es immer schwieriger macht, einen genauen Überblick über die eigentliche Funktionalität zu haben. Außerdem erschweren eine große Zahl von verschiedenen Netzwerkprotokollen oder andere Software- und Hardware-Komponenten, wie Firewalls und Network Address Translators (NATs), die Netzwerkanalyse. Zum Beispiel ist es immer schwieriger bestimmte Eigenschaften zu überprüfen, wie Reachability (Erreichbarkeit) und Slice Isolation (Isolation von Teilnetzen voneinander). Es gibt lediglich einige wenige Tools wie Ping oder Traceroute, die Netzwerkanalysten Hilfestellung leisten. Allerdings ist das oft nicht genug. Deshalb stellt Kazemian [1] ein neues Framework vor, das eine statische Netzwerkanalyse ermöglicht. Die sogenannte Header Space Analysis ermöglicht es ein Netzwerk nach häufigen Fehlerquellen zu durchsuchen. Zhang [3] zeigt allerdings, dass es sich dabei nicht um eine Innovation sondern lediglich um eine etwas umständlichen Anwendung von Finite State Machines (endliche Automaten) handelt. Dieses wird im Folgenden noch einmal aufgegriffen und die zwei Herangehensweisen werden miteinander verglichen.

Schlüsselworte

Header, Router, Netzwerkanalyse, Finite State Machine

1. EINLEITUNG

Auf praktische Weise kann man leicht in Computernetzwerken prüfen ob zwei Rechner miteinander kommunizieren können. Es wird lediglich ein Paket von A nach B geschickt. Solange das funktioniert, ist alles in Ordnung. Was aber wenn das Paket verloren geht oder möglicherweise sogar beim falschen Empfänger ankommt. Schon wird es schwierig den genauen Fehler zu finden. Die Größe von heutigen Netzen erschwert zunehmend die Netzwerkanalyse. Auf dynamischen Wegen ist diesem Problem oft fast nicht beizukommen. Daher beschäftigt sich diese Arbeit mit der statischen Analyse von Netzwerken. Man betrachtet nur ein unveränderliches Modell des Netzwerkes und versucht darin Fehler in den einzelnen Bausteinen zu finden.

Bei dieser Ausarbeitung wird zunächst in Kapitel 2 das Konzept der **Header Space Analysis** (HSA) vorgestellt und ein bestimmter Anwendungsbereich, der Überprüfung von Erreichbarkeit in Netzwerken, genauer betrachtet. Dabei wird auch die Komplexität des Algorithmus abgeschätzt und damit der Erfolg der HSA bei einem Test am Netz der Stanford University begründet. Anschließend wird in Kapitel 3 das gleiche Verfahren mit Hilfe von **Finite State Machines** (FSM) angewandt und ein Vergleich zur Header Space Analysis gezogen.

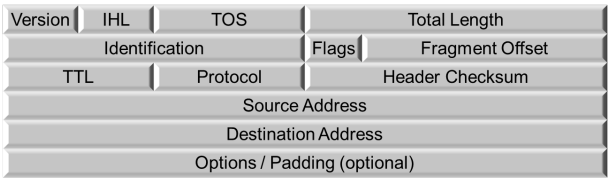


Abbildung 1: Aufbau eines IPv4-Headers. Darin sind alle wichtigen Informationen enthalten, um das Paket richtig zu bearbeiten. Zum Beispiel der Typ des Headers, seine Länge, der Typ des folgenden Headers, das Time-To-Live-Feld und natürlich Start- und Zieladresse.

2. HEADER SPACE ANALYSIS

Um verschiedene, teils komplizierte Abläufe in einem Netzwerk zu verstehen und mögliche Fehler aufzudecken, werden im Folgenden nur bestimme Netzwerke betrachtet. Diese verhalten sich unabhängig von der Zeit. Dabei bleiben sämtliche Verbindungen zwischen den Netzwerkknoten unverändert und die Funktionalität dieser ändert sich nicht. Dies nennt man statische Netzwerkanalyse.

Eine Methode dazu ist die Header Space Analysis. Dabei handelt es sich folglich um ein theoretisches Konzept, mit dessen Hilfe man versucht die Abläufe und Wege in einem Netzwerk zu analysieren. Anhand der Ergebnisse lassen sich anschließend mögliche Fehlerquellen finden. Der Name setzt sich zusammen aus der Analyse des mathematischen Raumes (Space), der von den Headerbits erzeugt wird.

Dabei wird versucht das reale Netzwerk so gut wie möglich in einem Model zu simulieren. Manche Informationen kann man dabei außen vor lassen, wie zum Beispiel die Checksumme oder, wenn man vor allem an IP-Routing interessiert ist, die Hardwareadressen. Andere sollte man auf jeden Fall beibehalten, da sie oft Ursache oder zumindest Indikator für häufige Probleme sind. Zum Beispiel das Time-to-Live Feld eines IP-Headers (vgl. Abbildung 1).

Pakete bestehen grundsätzlich aus zwei Teilen. Zum einen aus dem Header, in dem unter anderem alle relevanten Informationen des Senders und Empfängers enthalten sind, und zum anderen aus der Payload, dem Teil, der die eigentliche Nachricht enthält. Für den Weg des Pakets durch das Netzwerk ist die Payload irrelevant,daher wird im Folgenden nicht zwischen Header und Paket unterschieden.

doi: 10.2313/NET-2014-08-1_12

2.1 Definition des Frameworks

Die Header Space Analysis ist ein Model in dem folgende drei wichtigen Bestandteile eines realen Netzwerkes modelliert werden.

1. Als erstes werden sämtliche Pakete, die das Netzwerk passieren nur auf ihren Header reduziert. Diese werden unabhängig von der Art des Headers als Bitfolge einer bestimmten Länge, die man anfangs festlegen muss, dargestellt.

2. Zweitens wird der Ort eines Paketes, also die Position innerhalb des Netzes, als Port dargestellt. Ports sind eindeutig identifizierbar innerhalb des Netzwerkes und sind vergleichbar mit den Interfaces eines Routers. Zudem werden die Ports miteinander verbunden um die eigentliche Topologie des Netzwerkes zu simulieren.

3. Und drittens werden alle Netzwerkkomponenten, wie Router oder Switches, unabhängig ob es sich um Hardware- oder Softwarekomponenten handelt, als Middleboxes modelliert. Dabei ist es vollkommen egal was für eine Funktion sie haben. Es sind von außen gesehen nur Backboxes, die auf ankommende Pakte bestimmte Funktionen anwenden und sie dann weiterleiten. Der Verständlichkeit halber werden diese Middleboxes im Folgenden trotzdem mit Router oder Switches bezeichnet.

2.1.1 Header Space H

Der **Header Space** ist ein grundlegender Baustein des Frameworks. Alle möglichen Header einer festgelegten Länge werden als Vektoren, oder Punkte, in diesem Raum dargestellt. Die maximale Länge eines Headers muss anfangs festgelegt werden. Da Pakete oft unterschiedlich viele und unterschiedlich lange Header besitzen, je nach verwendeten Protokollen, sollte hier die maximale Länge gewählt werden. Kleinere Header können beispielsweise mit Padding-Bits aufgefüllt werden, da die meisten Header genaue Informationen enthalten, wie lang sie sind und welcher Header in der nächsten Schicht verwendet wird. Der Header Space ist mathematisch wie folgt definiert

$$H \subseteq \{0,1\}^L$$

wobei L die maximale Header-Länge ist. Man kann klar erkennen, dass in keiner Weise zwischen verschiedenen Headern oder Header-Feldern unterschieden wird, was eine Analyse unabhängig von sämtlichen Protokollen ermöglicht. Außerdem wird noch ein drittes Zeichen x, ein sogenanntes Wildcard-Zeichen, hinzugenommen, welches signalisiert, dass ein bestimmtes Bit beliebig ist, also sowohl 1 als auch 0 sein kann. Ein Header, der Wildcardbits enthält wird als Wildcardheader definiert. Man hat zwei Möglichkeiten einen Wildcardheader darzustellen. Entweder man erweitert den Header Space um dieses Zeichen.

$$H' \subseteq \{0,1,x\}^L$$

Oder man stellt sie als Vereinigung von normalen Headern dar, dabei wächst die Anzahl der Vereinigungen aber exponentiell zur Zahl der Wildcarzeichen. Ein Punkt im Header Space wird im Folgenden mit h bezeichnet.

2.1.2 Network Space N

Der **Network Space** setzt sich aus dem kartesischen Produkt des Header Space und der Menge aller Ports zusammen. Ports sind eindeutig identifizierbar in dem Netzwerk und immer mit einer Middlebox verbunden. Punkte im Network Space werden als Tupel (h,p) aus dem Raum

$$N \subseteq \{0,1\}^L \times \{0,...,P\}$$

dargestellt, wobei $\{0,...,P\}$ die Menge aller Ports innerhalb des Netzwerks ist. Dadurch lassen sich Pakete modellieren, die sich an einem bestimmten Ort (Port) im Netzwerk befinden. Der Network Space ist notwendig um zum Beispiel das Durchqueren eines Pakets durch ein Netzwerk zu modellieren, wobei es schrittweise von Router zu Router geschickt wird.
Die Identifikationsnummer des Ports auch binär kodiert und einfach an den Header angehängt. Somit handelt es sich bei einem Punkt im Network Space, im Vergleich mit einem Punkt im Header Space, um eine längere Bitfolge. Hier werden die Ports, aufgrund der besseren Lesbarkeit, entweder gemäß der Middlebox benannt, an der sie anliegen, zum Beispiel besitzt ein Router R die Ports R_0, R_1 und R_2. Oder sie werden einfach mit $p_1, p_2, ...$ durchnummeriert.

2.1.3 Transfer Funktion

Jeder Router, Switch, Firewall, Nat und alle anderen Netzwerkkomponenten werden vollkommen gleich als Middleboxes behandelt. Sie unterscheiden sich lediglich durch ihre Funktion für das Netzwerk. So unterscheidet sich zum Beispiel eine Firewall, die nur Pakete von bestimmten Absendern weiterleitet, von einem Router, der die Pakete anhand ihrer IP-Adresse in die richtige Richtung leitet, nur durch ihre individuelle Funktion. Diese werden in der Header Space Analysis als **Transfer Funktion** modelliert. Diese Funktionen lassen sich noch weiter untergliedern. Die grundlegendste davon ist die **Box Transfer Funktion**, welche die Funktion einer einzelnen Middlebox beschreibt. Mathematisch lässt sich die Funktion wie folgt beschreiben:

$$T(h,p) : (h,p) \rightarrow \{(h_1,p_1),(h_2,p_2),\dots\}$$

Bei dieser Definition fallen in erster Linie zwei Merkmale auf. Zum einen ist es möglich, dass sich der ankommende Header in seiner Struktur verändern kann. Dies geschieht zum Beispiel bei dem Time to Live (TTL) Feld in einem IPv4-Header oder mit der Source- und Destination-IP bei einer NAT-Middlebox. Zum anderen handelt es sich bei der Ausgabe der Funktion um eine Menge von Paketen, was in der Realität auch bei Broadcasts oder Multicasts geschieht. Zudem ist zu jeder Box Transfer Funktion auch eine Umkehrfunktion T^{-1} definiert. Allerdings nicht im mathematischen Sinne, da sie nicht injektiv ist. Festzuhalten ist allerdings, dass sich zu jeder Output-Menge gibt auch alle Header der Input-Menge bestimmen lassen. Die Signatur der Umkehrfunktion sieht also folgendermaßen aus

$$T^{-1} : \mathcal{P}(N) \rightarrow \mathcal{P}(N).$$

Mit $\mathcal{P}(N)$ ist die Potenzmenge von N bezeichnet, also der Menge, die alle verschiedenen Teilmengen aus dem Netzwerk Space enthält. Insbesondere enthält sie auch die leere Menge.

Um die Transfer Funktion besser zu verstehen folgt ein kurzes Beispiel. Ein Router, der ankommende Pakete anhand

doi: 10.2313/NET-2014-08-1_12

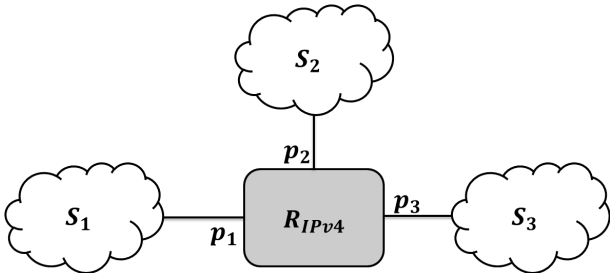


Abbildung 2: Ein IPv4-Router, der mit drei verschiedenen Netzen an den jeweiligen Ports verbunden ist

ihrer Ziel-IP-Adresse in die jeweiligen Subnetze weiterleitet, wie in Abbildung 2 zu sehen, kann mit folgender Transferfunktion modelliert werden. Dabei ist die Funktion $ip_dest(h)$ eine Hilfsfunktion, die die Ziel-IP-Adresse des Headers h zurückliefert.

$$T_{IPv4}(h,p) = \begin{cases} \{(h,p_1)\} & \text{if } ip_dest(h) \in S_1 \\ \{(h,p_2)\} & \text{if } ip_dest(h) \in S_2 \\ \{(h,p_3)\} & \text{if } ip_dest(h) \in S_3 \\ \{\} & \text{otherwise} \end{cases}$$

Da sich Netzwerke in der Regel aus vielen Routern, Switches, usw. zusammensetzen, wird eine Hilfsfunktion eingeführt, die alle Middleboxes innerhalb des Netzes simuliert. Sie hat die gleichen Ein- und Ausgaben wie die normale Box Transfer Funktion und dient lediglich als Hilfe um die Anwendungen übersichtlicher zu gestalten. Seien $T_1, ..., T_n$ alle Box Transfer Funktionen der jeweiligen Switches $switch_1$, ..., $switch_n$ innerhalb des Netzwerkes, dann ist die sogenannte **Network Transfer Funktion** Ψ demnach folgendermaßen definiert:

$$\Psi(h,p) = \begin{cases} T_1(h,p) & \text{if } p \in switch_1 \\ ... & ... \\ T_n(h,p) & \text{if } p \in switch_n \end{cases}$$

Diese Funktion ruft die zu dem aktuellen Port passende Box Transfer Funktion auf. So genügt ein einzelner Aufruf der Ψ-Funktion, um die jeweilige Transferfunktion aufzurufen, unabhängig von der Position des Pakets im Netzwerk.
Bisher wurde noch in keinem Schritt die tatsächliche Topologie des Netzwerkes modelliert. Das ist die eigentliche Struktur, die festgelegt welche Middleboxes miteinander verbunden sind. Dabei werden meistens einfache, ungerichtete Graphen verwendet, auf denen man die üblichen Graphenalgorithmen ausführen kann, wie das Finden von Pfaden. Dazu wird neben der Liste der Ports auch noch eine Menge von Tupeln von der Form (p_i, p_j) benötigt, die bedeuten, dass Port i mit Port j verbunden ist. Im Weiteren wird einfachheitshalber davon ausgegangen, dass alle Verbindungen bidirektional sind, also in beide Richtungen gehen. Die **Topology Transfer Funktion** Γ ist so definiert, dass sie das Eingabepaket gemäß dem Netzwerkgraphen an den nächsten Port weiterleitet.

$$\Gamma(h,p) = \begin{cases} \{(h,p^*)\} & \text{if p is connected to } p^* \\ \{\} & \text{if p is not connected} \end{cases}$$

Sie modelliert das Verhalten sämtlicher Links oder Verbindungen in dem Netz. Dabei ist zu beachten, dass sich nur der Port ändert und die Header unverändert bleiben. Außerdem ist zu dieser Funktion die Umkehrfunktion analog definiert, da Netzwerke hier als ungerichtete Graphen betrachtet werden und die Verbindungen bidirektional sind. Es gilt also immer $\Gamma^{-1} = \Gamma$.

Die Network Transfer Funktion und die Topology Transfer Funktion werden immer nacheinander aufgerufen, um einen Hop eines Paketes zu simulieren. Daher wird die Konkatenation dieser beiden Funktionen in einer eigenen Funktion zusammengefasst, nämlich $\Phi(.) = \Psi(\Gamma(.))$. Der konsekutive Aufruf dieser Funktion ermöglicht mehrere Hops oder entsprechende Schritte in einem Netzwerk zu gehen. Die folgende Funktion ermittelt alle Tupel (h', p'), die erzeugt werden, wenn man mit (h,p) startet und k Schritte berechnet.

$$\Phi^k(h,p) = \Psi(\Gamma(...\Psi(\Gamma(h,p)...)$$

Damit sind die wichtigsten Bestandteile des Header Space Frameworks definiert und man kann damit beginnen, erste Netzwerkprobleme zu analysieren.

2.2 Reachability

Eine der grundlegendsten Fragen in der Netzwerkanalyse ist „Kann Host a Pakete zu Host b senden?“, oder „Welche Pakete erreichen Host b von Host a?“. Diese Frage lässt sich eindeutig beantworten.
Der dafür verwendete Algorithmus lässt sich in zwei Hauptbestandteile zerlegen. Zunächst berechnet man durch mehrmaliges Anwenden der Φ-Funktion auf einen Wildcardheader die Menge der Header, die b erreichen um anschließend durch Anwenden der Umkehrfunktionen die ursprünglichen Header, die von Host a losgeschickt werden, zu erhalten.

2.2.1 Algorithmus

Diese erste Idee bedarf aber noch einiger Feinheiten, um die Komplexität zu reduzieren. Zum Beispiel würden durch Anwenden der Φ-Funktion auch Pakete in Netzwerkbereiche weitergeleitet werden, von denen man den Ziel-Host gar nicht mehr erreichen kann. Deshalb werden anfangs alle Pfade von a nach b aufgelistet. Allerdings stellt sich dabei die Frage, wie mit Kreisen im Netzwerkgraph verfahren werden soll. Ein mögliches Vorgehen wäre es, eine maximale Anzahl von Schleifen festzusetzen und somit alle Pfade aufzulisten. Dabei genügt logischerweise eine geringe Zahl, da Netzwerke auf Effizienz ausgerichtet sind und es nicht passieren sollte, dass sich Pakete im Kreis bewegen.
Nun schickt Host a einen kompletten Wildcardheader, also einen Header, in dem jedes Bit beliebig ist, los und man wendet darauf alle Transferfunktionen entlang des aktuellen Pfades an. Anschließend wird dies für jeden Pfad gemacht und die jeweiligen Mengen vereinigt. Sei $R_{a\to b}$ die Menge aller Header, die b von a erhält und sind die Pfade von der Form

$$a \to S_1 \to ... \to S_{n-1} \to S_n \to b$$

so erhält man für die Menge der bei b ankommenden Header

$$R_{a\to b} = \bigcup_{a\to b \text{ paths}} T_n(\Gamma(T_{n-1}(...(\Gamma(T_1(h,p)...)).$$

Nach diesem Schritt muss man nun den ganzen Weg wieder zurück gehen, da sich diese Header auf dem Weg geändert

doi: 10.2313/NET-2014-08-1_12

haben können. Demnach ist es notwendig durch Anwenden der Umkehrfunktionen die ursprüngliche Header-Menge h_a zu berechnen.

$$h_a = T_1^{-1}(\Gamma(...(T_{n-1}^{-1}(\Gamma(T_n^{-1}(h,b)...))$$

Damit ist die Frage der Erreichbarkeit beantwortet. Allerdings liefert der Algorithmus zunächst nur die Information, dass Pakete ankommen oder nicht. Er liefert nicht die möglichen Fehlerursachen. Will man diese aufspüren, kann man zum Beispiel das Netz nach dem Divide-and-Conquer Prinzip in kleinere Teilnetze zerteilen und den Algorithmus auf diese ausführen.

Zum besseren Verständnis findet man in Abbildung 3 ein Beispiel. Dabei werden Header der Länge acht bit verwendet. Zur Visualisierung der Header an den jeweiligen Positionen werden sie in ein Koordinatensystem eingetragen, wobei jeweils die ersten vier Bits, als natürliche Zahl interpretiert, in x-Richtung und die letzten vier in y-Richtung abgetragen werden. Das Ziel ist wiederum alle Header zu finden, die b von a erreichen. Alle Header die in den einzelnen Schritten des Algorithmus entstehen sind über bzw. unter die jeweiligen Router als Graph dargestellt. Die Transferfunktionen der Middleboxes lauten

$$T_A(h,p) = \begin{cases} \{(h, A_0)\} & \text{if } h = 1001xxxx \\ \{(h, A_1)\} & \text{if } h = 1010xxxx \\ \{(h, A_2)\} & \text{if } h = 1100xxxx \end{cases}$$

$$T_B(h,p) = \begin{cases} \{(h, B_0)\} & \text{if } h = 10xxxx01 \\ \{(h, B_1)\} & \text{if } h = 10xxxx00 \end{cases}$$

$$T_C(h,p) = \begin{cases} \{((h\&00001111)|11000000, C_0)\} & \text{if } h = 0011xxxx \\ \{((h\&00001111)|00110000, C_1)\} & \text{if } h = 1100xxxx \end{cases}$$

$$T_D(h,p) = \begin{cases} \{(h, D_0)\} & \text{if } h = xxxx1xxx \\ \{(h, D_1)\} & \text{if } h = xxxx0xxx \end{cases}.$$

Die Funktionen T_A, T_B und T_D gehören zu einfachen Middleboxes, die in Abhängigkeit von bestimmten Header-Feldern Entscheidungen treffen, wie zum Beispiel Router oder Firewalls. Dagegen wirkt T_C auf den ersten Blick komplizierter. Grundsätzlich erkennt man, dass sie nicht nur Pakete weiterleitet sondern auch ihre Header verändert. Sie ändert die ersten vier Bits von 1100 zu 0011 und umgekehrt. Dies ist durch ein binäres AND mit einer Maske und anschließendem binärem OR implementiert. Die Funktionsweise von T_C ähnelt der eines Network Address Translators (NAT).
Die Menge der Header, die in b ankommen, wird zunächst durch Anwenden der jeweiligen Transferfunktionen berechnet, wie in Abbildung 3 visualisiert. Die Anzahl der einfachen Pfade in diesem Netzwerk ist Zwei und es wird auf das Hinzunehmen von Schleifen verzichtet. Man erhält also

$$R_{a\to b} = \{10100x01, 00110xxx\}.$$

Anschließend wendet man die Umkehrfunktionen an und erhält

$$h_a = \{10100x01, 11000xxx\}.$$

Offensichtlich ist h_a zum Teil identisch mit $R_{a\to b}$. Je nachdem welche Header-Felder man modelliert, ist dies unterschiedlich. Zum Beispiel das TTL Feld garantiert, dass sie nicht identisch sind. Das Anwenden der Umkehrfunktion ist also auf jeden Fall notwendig.

2.2.2 Komplexität

Als Erstes wird versucht die Worst-Case Laufzeit abzuschätzen. Dabei kann jeder Router $O(2^L)$ neue Pakete erzeugen. Diese kann er jeweils an jeden seiner Ports schicken. Wenn das Netzwerk einen Durchmesser d hat und die genannten Fälle an jedem Router eintreten erhält man

$$O((2^L * P)^d) \text{ mit } P = |ports|.$$

Zudem kann d noch weiter wachsen indem Schleifen beliebiger Länge auftreten.

Allerdings überschätzt diese Schranke die tatsächliche Laufzeit deutlich, da die Header Space Analysis in der Praxis durchaus gute Ergebnisse liefert, wie das nächste Kapitel zeigt. Unter Hinzunahme einer Annahme über die Struktur von Netzwerken, kann deshalb die Komplexität auf durchschnittlich $O(dR^2)$ reduziert werden.
Es wird angenommen, dass Computernetze immer auf Effizienz ausgerichtet sind. Pakete sollen möglichst schnell und direkt übertragen werden. Demnach sollen auf ein eingehendes Paket nur sehr wenige Transferregeln zutreffen. Sei R_1 die Anzahl der Pakete die an einem Router ankommen und R_2 die Zahl der Transferregeln des Routers. Man geht davon aus, dass anstatt R_1R_2 nur cR_1, mit $c << R_2$, neue Header erzeugt werden. Diese Annahme wird als **Linear Fragmentation Assumption** bezeichnet. An einem Router werden $O(R)$ ankommende Header mit $O(R)$ Transferregeln verglichen und generieren wiederum $O(R)$ Header. Demnach liegt die Komplexität, wie bereits erwähnt, in

$$O(R^2 + R^2 + ... + R^2) = O(dR^2).$$

Diese Annahme findet in der Praxis Bestätigung, da das Framework an einem größeren Netzwerk, nämlich dem Netz der Stanford University, erfolgreich getestet wurde.

2.3 Hassel und ein Test am Stanford Netz

Auch wenn die Header Space Analysis in der Theorie zu funktionieren scheint, ist das noch keine große Hilfe zur praktischen Analyse von Netzwerken. Daher wurde, neben der theoretischen Grundlage, auch ein Tool namens **Header Space Library** oder **Hassel** implementiert. Dieses in Python geschriebene Programm kann unter anderem aus Routern die entsprechenden Transferfunktionen auslesen und sie in eine für die *Header Space Analysis* kompatible Form bringen. Allerdings ist das bis jetzt nur bei Cisco-Routern möglich.
Anschließend stellt Hassel mehrere Funktionen zur Netzwerkanalyse bereit, wie die schon genannte Überprüfung auf Erreichbarkeit und zum Beispiel das Suchen nach Schleifen im Netzwerk, oder das Überprüfen ob ein Teilnetz von einem anderen isoliert ist, was im Sinne der IT-Sicherheit interessant ist. Schließlich wurde Hassel am Netz der Stanford University getestet, einem Netz bestehend aus unter anderem fünf /16 IPv4 Subnetzen und über 757,000 Forwarding-Regeln. Der Test fand unter anderem 12 Endlosschleifen, wobei die TTL außer acht gelassen wurde. Genaue Ergebnisse sind in Kazemias Arbeit [1] nachzulesen. Festzuhalten ist, dass man trotz großer Datenmenge in relativ kurzer Zeit

doi: 10.2313/NET-2014-08-1_12

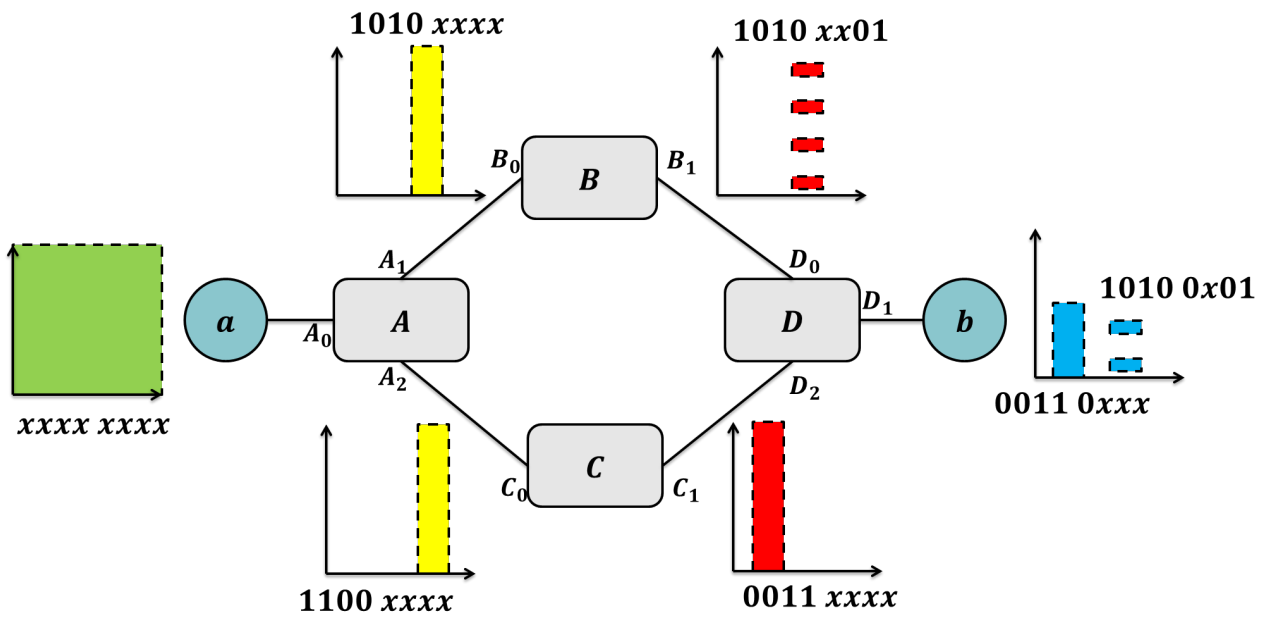


Abbildung 3: Überprüfen der Erreichbarkeit zwischen den Hosts a und b mithilfe der HSA. Hier ist visualisiert, wie die Header ihre Form verändern auf ihrem Weg von a nach b.

vorzeigbare Ergebnisse erhalten hat.
Da nun die Header Space Analysis definiert und ihre Funktion und Effizienz aufgezeigt wurde, kann nun der Vergleich mit den Finite State Machines gezogen werden.

3. FINITE STATE MACHINES

Der Ansatz der Header Space Analysis wirkt auf den ersten Blick wie eine Innovation, bei genauerem Betrachten ist er allerdings einem Verfahren mit **Finite State Machines** (FSM), auf deutsch Endlichen Automaten, sehr ähnlich. Endliche Automaten finden vor allem in der Theoretischen Informatik Anwendung. Zhang bezeichnet den Ansatz der Header Space Analysis als isomorph zu dem Ansatz mit Finite State Machines [3].

3.1 Definition Finite State Machine

Bei einer Finite State Machine handelt es sich grundsätzlich um ein 5-Tupel $(\Sigma, S, s_0, \delta, F)$. Dabei ist Σ eine endliche, nicht leere Menge von Zeichen, genannt das Eingabealphabet. S ist eine endliche, nicht leere Menge von Zuständen und s_0 ein Element aus S, das der Startzustand ist. Je nach Anwendungsgebiet ist es auch möglich, eine Teilmenge von S als Startzustand festzulegen. Der wichtigste Bestandteil des Automaten ist die Übergangsfunktion $\delta : \Sigma \times S \rightarrow P(S)$. Sie erzeugt eine Menge von Zuständen in Abhängigkeit des aktuellen Zustandes und der Eingabe. Schließlich gibt es auch noch eine Menge F von Endzuständen, den Zuständen in denen der Automat stoppt.
In Abbildung 4 sieht man eine graphische Darstellung eines endlichen Automaten. Dabei werden Zustände als Kreise, die Übergangsfunktion als Pfeile zwischen den Kreisen, das Eingabealphabet als Zeichen über den Pfeilen und die Enzustände als Doppelkreis dargestellt.

3.2 Reachability mit Finite State Machines

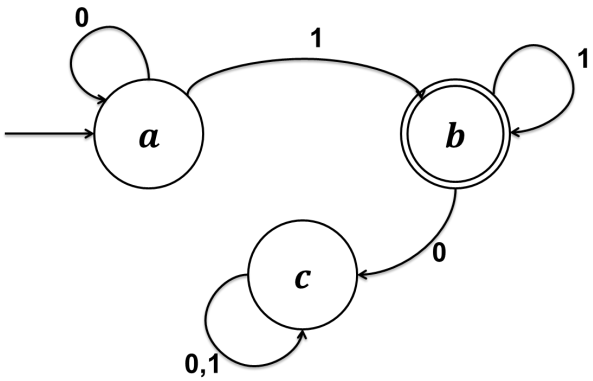


Abbildung 4: Graphische Darstellung eines endlichen Automaten mit Zuständen {a,b,c}. Die Übergangsfunktion ist mit Pfeilen visualisiert, an denen das jeweilige Zeichen aus dem Eingabealphabet steht. Dieser einfache Automat kann zum Beispiel zum erkennen der Sprache 0^*11^* verwendet werden

Ein Automat, der die gleiche Funktionalität wie die *Header Space Analysis* aufweist, lässt sich folgendermaßen konstruieren. Die Menge der Zustände ist die Menge aller Punkte aus dem Network Space $\{0,1\}^L \times \{0, ..., P\}$. Auf den ersten Blick wirkt das nicht sehr effizient, da allein die Zustandsmenge Ausmaße der in der Größenordnung $O(2^{L+P})$ annimmt. Das ist ein allgemeines Problem bei Automaten mit großer Zustandsmenge. Daher wird die Zustandsmenge meistens nicht explizit angegeben und gehofft bzw. vorhergesagt, dass die Menge der aktuellen Zustände nicht explodiert, also rapide ansteigt. Als Eingabealphabet kann man je nach Anwendung verschieden Mengen wählen. Für das

doi: 10.2313/NET-2014-08-1_12

Überprüfen der Erreichbarkeit genügt irgendeine einelementige Menge, deshalb wird für diesen Algorithmus das Eingabealphabet außer Acht gelassen.
Als Übergangsfunktion wird die aus der HSA schon bekannte Φ-Funktion verwendet. Diese verbindet in der State Machine alle Zustände, die in dem Netzwerk in einem Hop erreicht werden können. Im Bezug auf die Erreichbarkeit ist auch klar welche Mengen man für den Start- und Endzustand wählen muss, nämlich zum einen die Menge aller Zustände der Form $(..., a)$ und zum anderen alle Zustände der Form $(..., b)$. Also alle Pakete, die bei Port a beziehungsweise bei Port b liegen.

3.3 FSM Algorithmen

Die meisten Algorithmen, die Erreichbarkeit auf endlichen Automaten überprüfen, arbeiten nach einem einfachen Prinzip. Es werden rekursiv Zustände hinzugefügt, die von den aktuellen Zuständen erreicht werden können. Diese Iteration lässt sich folgendermaßen formulieren. Sei R_i die Menge von Zuständen, die in bis zu i Schritten erreicht werden können, dann ist

$$R_n = R_{n-1} \cup \bigcup_{y \in R_{n-1}} T(y).$$

Die Menge R_n setzt sich also aus zwei Teilen zusammen. Zum einen allen Zuständen, die in $n-1$ Schritten erreicht werden, und zum anderen allen Zuständen, die von der aktuellen Menge durch Anwenden der Transferfunktion erreicht werden können. Dieser Schritt wird anschließend solange wiederholt bis ein Fixpunkt R^* erreicht wird. Also sich die Menge der erreichten Zustände nicht mehr ändert. Dies ist der Fall, wenn gilt

$$R^*(x) = R_n = R_{n-1}.$$

Diese Fixpunktiteration wird als **Structural FSM Traversal** bezeichnet [2]. Auf den ersten Blick scheint dieser Algorithmus ein sicherer Weg zu allen erreichbaren Zuständen zu sein, allerdings wird dabei ein großes Problem außer Acht gelassen. Es kann nämlich passieren, dass die Zustandsmenge explodiert und somit eine effiziente Berechnung unmöglich macht. Dies ist zum Beispiel dann der Fall wenn das Netzwerk unendliche Kreise enthält und man diese nicht vorher herausfiltert. Dennoch liegt in der Regel die Anzahl der Zustände bei diesen Automaten in Bereichen, mit denen man nicht mehr effizient arbeiten kann. Zum Beispiel erhält man bei einer Headergröße von 32 bit und einem Netzwerk mit nur acht Ports bereits bis zu $2^{40} \approx 1.0 \cdot 10^{12}$ verschiedene Zustände.
Man ist schließlich darauf angewiesen, dass die Zustandsmenge in geordneten Bahnen wächst. Außerdem ist es wichtig die erreichten Zustände in geeigneten Datenstrukturen abzulegen, um ein effizienteres Arbeiten zu ermöglichen. Dabei werden häufig **Binary Decision Diagramms** (BDDs) oder **Disjunktive Normalformen** (DNFs) verwendet. Es ist immer wichtig die passende Datenstruktur für das jeweilige Problem auszuwählen.
Unabhängig davon weist diese Iteration starke Ähnlichkeiten zur HSA auf. Tatsächlich handelt es sich bei der Header Space Analysis um einen Ansatz der auf Finite State Machines basiert.

3.4 Vergleich FSM mit HSA

Wie bereits erwähnt lässt es sich zeigen, dass es sich bei der Header Space Analysis um ein Verfahren handelt, das sehr stark an Finite State Machines angelehnt ist. [3] Dabei werden die jeweiligen Headerbits in disjunktiver Normalform angegeben. Zum Beispiel der Header $1x01$ wird als der boolesche Ausdruck

$$x_0 \wedge \neg x_2 \wedge x_3$$

repräsentiert. Wendet man die Fixpunktiteration auf das Beispiel aus Abbildung 3 an, erhält man schrittweise den Baum aus Abbildung 5, ähnlich wie bei der Breitensuche in normalen Graphen. Die Bezeichnung an den Kanten wurde weggelassen, da das Eingabealphabet hier nur aus einem Zeichen besteht. Es ist gut erkennbar was passieren würde, wenn jeder Knoten viele weitere erzeugen würde. Die Zahl würde exponentiell wachsen und eine weitere Berechnung sehr verlangsamen beziehungsweise unmöglich machen. Außerdem kann man wie schon bei der Header Space Analysis unwichtige Zustände ausblenden, indem man zunächst die Pfade im Netzwerk durchnummeriert und anschließend nur diesen Pfaden folgt.

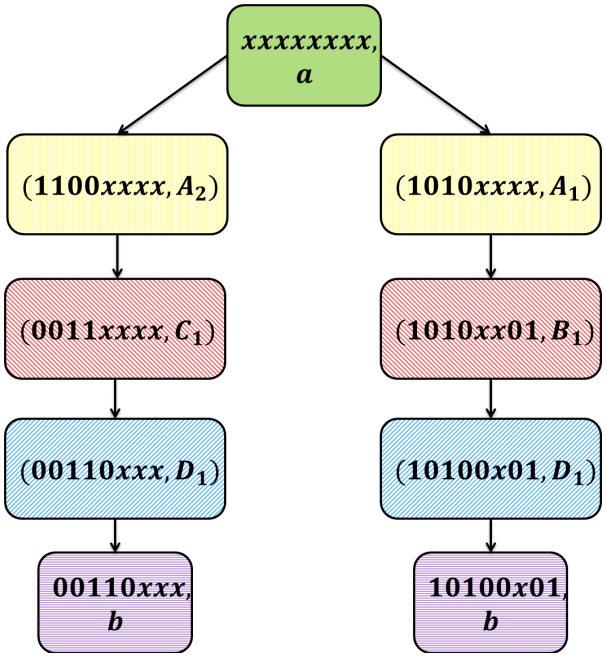


Abbildung 5: Anwendung der Fixpunktiteration auf das vorherige Beispiel. Die jeweilig erreichten Zustände erzeugen einen Baum, wobei die Tiefe maximal der Anzahl der Schritte entspricht. Gleiche Knoten können entfernt werden. Jeweils gleiche Farben werden bei gleichem Iterationsschritt erreicht.

Das wurde in Abbildung 5 bereits so gemacht, da ansonsten der Baum bereits bei diesem sehr kleinen Netzwerk komplizierter werden würde. Wenn alle erreichten Zustände nach ihren jeweiligen Ports zusammengefasst werden, also unter anderem alle Zustände, die einen Port enthalten, der zu D gehört, erhält genau die gleiche Aufstellung, die man auch mit der eader Space Analysis erhalten hat. Außerdem gelten für die Finite State Machines auch die gleichen Annahmen, wie zum Beispiel die Linear Fragmentation Assumption. So kann man in der Regel bei einem Computernetzwerk davon ausgehen, dass die Zustandsmenge nicht übermäßig wächst.

doi: 10.2313/NET-2014-08-1_12

Neben dem Erreichbarkeitsproblem lassen sich auch andere Probleme, die in der Header Space Analysis gelöst wurden, genauso mithilfe von Finite State Machines bewältigen. Das Problem der Loop-Erkennung wurde in der HSA [1] unnötig verkompliziert, und kann mithilfe allgemeiner Algorithmen auf Finite State Machines [3] gelöst werden.
Die Simulation von FSMs ist bei wachsender Komplexität immer schwieriger, da die Zustandsmenge zu schnell explodiert. Allerdings ist der unerwartete Erfolg der Header Space Analysis unumstritten und der gezeigte Isomorphismus zu Finite State Machines führt dazu, das man mit ihnen ähnlich gute Ergebnisse erzielen kann, und wahrscheinlich sogar auf effizienteren Wegen. Da es für die Simulation von Automaten optimierte Software gibt, die über die Jahre immer weiter optimiert wurde.

4. SCHLUSS

Nachdem die Header Space Analysis bei den Tests am Stanford Netz gute Resultate lieferte, kann man davon ausgehen, dass die getroffenen Annahmen zumindest zum Teil der Wirklichkeit entsprechen, unter denen die Komplexität wesentlich geringer ist als die gezeigte Worst-Case Laufzeit war. Aber wenn schon die, umständlichen und schwächeren Techniken zu ansehnlichen Resultaten führen [3], dann ist es wahrscheinlich möglich mit Finite State Machine Algorithmen bessere Resultate zu erzielen. Nichtsdestotrotz funktioniert der Ansatz der Header Space Analysis, auch wenn es sich dabei im Großen und Ganzen um eine FSM für Computernetzwerke handelt. Zudem sollte man die Methoden noch gegen größere Netzwerke testen, um die Stabilität des Verfahrens genauer zu prüfen, da die Effizienz bislang nur mit Annahmen über die Netzwerke und nicht über die Algorithmen begründet wurde. Alles in allem lässt der unerwartete Erfolg der Header Space Analysis darauf hoffen, dass man sich wieder mehr mit der formalen Verifikation von Finite State Machines für Computernetzwerken beschäftigt.

5. LITERATUR

[1] Peyman Kazemian, George Varghese, and Nick McKeow. Header space analysis: Static checking for networks. *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, 2012.

[2] Dominik Stoffel, Markus Wedler, Peter Warkentin, and Wolfgang Kunz. Structural fsm traversal. *IEEE Transactions on Computer-Aided Design of Integrated Circuits ad Systems*, 23(5), 2004.

[3] Shuyuan Zhang, Sharad Malik, and Rick McGeer. Verification of computer switching networks: An overview. *Automated Technology For Verification and Analysis*, 2012.

doi: 10.2313/NET-2014-08-1_12

Northbound API Requirements for SDN

Christian Fuchs
Betreuer: Daniel Raumer
Seminar Innovative Internettechnologien und Mobilkommunikation SS2014
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: fuchs@in.tum.de

KURZFASSUNG

Software Defined Networking (SDN) hat über die letzten Jahre eine starke Verbreitung erfahren. An der Schnittstelle zwischen den Netzwerkgeräten und den Controllern, dem sogenannten Southbound Interface, hat sich das Protokoll OpenFlow als ein Standard durchgesetzt, der von großen Teilen des Markts akzeptiert und unterstützt wird. An der Schnittstelle zwischen den Contoller Frameworks und den Applikationen, dem Northbound Interface (NBI), verhält sich dies jedoch grundlegend anders, da hier jedes Controller Framework eine eigene API (Application Programming Interface) besitzt. Aufgrund des Umstands, dass die Anwendungsgebiete von SDN sehr vielseitig sind, unterscheiden sich diese APIs teilweise stark voneinander. Das bringt einige Probleme mit sich, weshalb der Wunsch nach einem Standard für diese Northbound API immer stärker wird. Doch auch der Prozess der Standardisierung ist mit Schwierigkeiten verbunden, was zur Folge hat, dass derzeit noch kein Standard existiert und bis jetzt auch noch nicht abzusehen ist, ob und in welcher Form ein solcher Standard etabliert werden könnte.

Schlüsselworte

Software Defined Networking, Northbound API, Northbound Interfaces, Southbound Interface, NBI Standardisierung, Pyretic, SDN Controller, OpenFlow

1. EINLEITUNG

2008 wurde von einer Gruppe von Forschern der Universitäten Stanford, Princeton, Washington, dem MIT und der UC California im Artikel „OpenFlow: Enabling Innovation in Campus Networks“ [15] das Protokoll OpenFlow vorgestellt. Dieses sollte, aufbauend auf einem Netz aus OpenFlow fähiger Hardware, ermöglichen, Experimente im universitätseigenen Netzwerk durchzuführen, ohne den normalen Netzbetrieb zu stören oder großen Konfigurationsaufwand betreiben zu müssen. Das Konzept, das dieser vorgestellten Technik zugrunde lag, in anderer Form bereits durch andere Projekte, wie zum Beispiel RFC 5810 „Forwarding and Control Element Separation“ behandelt und unter dem Namen Software Defined Networking bekannt wurde, bot jedoch auch einen Lösungsansatz für Probleme, die weit über die zu Anfang adressierte Problematik hinausging, was großes Interesse vonseiten der Netzwerkindustrie hervorrief. Die darauffolgende Verbreitung von SDN wird häufig als wahrer „Hype“ beschrieben und führte dazu, dass SDN schon wenige Jahre später in die meisten Gebiete der Netzwerktechnik Einzug gehalten hat, was, wenn man die Dauer anderer Innovationen im Netzwerkumfeld bedenkt, wie beispielsweise der Einführung von IPv6, durchaus bemerkenswert ist. Es ist aber auch klar, dass diese rasante Entwicklung herausfordernde Probleme mit sich bringt. Eines dieser Probleme, welches aktuell viele Akteure im SDN Umfeld beschäftigt, ist die Diskussion um die Standardisierung des NBIs. Bevor jedoch auf diese Problematik eingegangen wird, soll in dieser Arbeit zunächst in Kapitel 2 das Konzept und die Anwendungsgebiete des Software Defined Networkings kurz vorgestellt werden, um eine Grundlage für die folgenden Abschnitte zu schaffen. Im dritten Abschnitt sollen daraufhin die Begriffe Northbound und Southbound Interface eingeführt, unter Punkt 3.1 kurz auf das Southbound Interface OpenFlow eingegangen und unter Punkt 3.2 die Probleme am NBI thematisiert werden. Punkt 4 beinhaltet das Kernthema dieser Arbeit. Es sollen dort insbesondere mögliche Anforderungen an eine Northbound API, die Vor- und Nachteile einer Standardisierung, sowie die Anforderungen an einen NBI Standard herausgearbeitet werden. Abschnitt 4.6 beschäftigt sich kurz mit den Unterschieden im Standardisierungsprozess von Northbound und Southbound Interface. Im letzten Abschnitt soll abschließend als Beispiel für ein relativ abstraktes und mächtiges NBI Pyretic beschrieben werden. Hier konzentriert sich diese Arbeit auf Modularisierung und Kompositionsoperatoren, das abstrakte Paket Modell, die High-Level Policy Funktionen und die Grenzen, denen Pyretic unterworfen ist.

2. SOFTWARE DEFINED NETWORKING - KONZEPT UND ANWENDUNGEN

Das grundlegende Modell von SDN ist unkompliziert und wurde bereits in vielen wissenschaftlichen Arbeiten ausführlich beschrieben. Einen guten Einstieg bietet ein Whitepaper der Open Networking Foundation (ONF) [18] oder der Artikel [10]. Sehr gut werden die Komponenten und Schlüsselprinzipien von SDN auch in [21], einem weiteren Dokument der ONF, beschrieben. Hier ist auch das SDN Grundprinzip prägnant zusammengefasst: „The control plane is (1) logically centralized and (2) decoupled from the data plane“ ([21], Seite 5). Dabei bezeichnet die Control Plane die Ebene in der Paketverarbeitung, auf der die Entscheidungen bezüglich der Weiterleitung von Paketen getroffen wird. Ein Teil dieser Control Plane kann beispielsweise die Routingtabelle sein. Die Data Plane ist im Gegensatz dazu konkret für die Weiterleitung der Pakete verantwortlich. Kommt also ein Paket an einem Netzwerkgerät an, wird die Information wie es behandelt werden soll aus der Control Plane geholt und die Data Plane verarbeitet das Paket dann gemäß dieser In-

doi: 10.2313/NET-2014-08-1_13

formationen, passt also zum Beispiel Header Felder an und gibt es auf einem bestimmten Interface wieder aus, droppt es oder leitet es an eine bestimmte Applikation weiter. Der Begriff **Software Defined** Networking ist darauf zurückzuführen, dass der Controller, in welchen die Control Plane ausgegliedert wird, in der Regel in Software realisiert ist. Dies hat zur Folge, dass sämtliche Kontrollinformation, also die gesamte Netzintelligenz, in Software vorliegt. Damit kontrolliert die Software das gesamte Netzwerkverhalten, das Netzwerk ist also durch Software „definiert".
Doch inwiefern lässt sich dieses Konzept auf reale Systeme übertragen? Eine in der Realität häufig auftretende Topologie besteht aus einer Menge an Hosts, die durch miteinander konkatenierten, programmierbaren Switches verbunden sind. In traditionellen Netzwerken ohne SDN muss zur Beeinflussung des Netzwerkverhaltens jeder Switch einzeln konfiguriert werden, da hier innerhalb eines Switches die Control Plane fest mit der Data Plane verbunden ist. Dies bringt diverse Probleme mit sich: Zum Beispiel ist selbst bei sorgfältig durchgeführter Konfiguration die Gefahr von Inkonsistenzen sehr hoch, was besonders in sicherheitsrelevanten Systemen, wie Firewalls oder Access Control Mechanismen, sehr gefährlich sein kann. Man hat auch keine globale Sicht auf das Netzwerk, was beispielsweise die Realisierung von komplexeren Quality of Service (QoS) Mechanismen stark erschwert. Ein weiteres Problem sind die Konfigurationssprachen an sich. Diese sind zum einen meist sehr hardwarenah, bieten also nur wenig Abstraktion, und sind zum andern häufig herstellerspezifisch, was den herstellerübergreifenden Tausch von Hardware bei vorhandener Konfiguration wesentlich verkompliziert. Folglich sind solche Topologie mit hohem Aufwand und hohen Kosten bei Entwicklung und Administration der Netzwerke verbunden, insbesondere da auch Skalierbarkeit eine große Problematik darstellt. Denn bei steigender Anzahl an Netzwerkgeräten verschärfen sich die aufgeführten Probleme zunehmend.
Hier schafft SDN Abhilfe. Durch die zentrale Konfigurationsschnittstelle am Controller, welche meist eine globale Sicht auf das Netzwerk und, abhängig von der Gestaltung der Northbound API, oft auch ein höheres Abstraktionslevel bietet, lösen sich viele der beschriebenen Probleme, wie zum Beispiel die Skalierbarkeitsproblematik, von selbst. Bei Einsatz von standardisierten Protokollen wie OpenFlow, ist sowohl der herstellerübergreifende Wechsel der Hardware, als auch die Realisierung von Multivendor-Umgebungen problemlos möglich. Diese und weitere Vorteile von SDN sind insbesondere bei der Verwirklichung von Virtualisierungstechniken, welche ständig weiter an Bedeutung gewinnen, überaus wichtig. Und der Bedarf an mächtiger und dennoch einfach zu konfigurierender Virtualisierung ist aktuell schon sehr groß, bedenkt man zum Beispiel all die Produkte im Feld der mobilen Endgeräte, von Cloud Services, von Cloud Computing und von Infrastrukture as a Service (IaaS) Diensten, bei denen Endkunden virtuelle Maschinen in virtuellen Netzen verschiedenster Art und Form angeboten werden. Virtualisierung und die SDN Lösungen, auf denen diese basiert, sind schon heute in vielen Bereichen unverzichtbar geworden und werden folglich auch in sämtlichen Bereichen der Netzwerkwelt verwendet. Einige populäre Beispiele hierfür sind das SDN basierte Wide Area Network (WAN) G-Scale von Google, welches weltweit Datenzentren miteinander verbindet [7], Googles interne Virtualisierungstechnik Andromeda [27], die seit April 2014 auch Googles Cloud Service Nutzern zur Verfügung steht, oder Microsofts Hyper-V Technologie, die Virtualisierungstechnik in Microsoft Server 2012 [22].
Die vielseitige Verwendung von SDN hatte nicht nur eine starke Verbreitung des Konzepts zur Folge, sondern auch, dass sich die Bedeutung des Begriffs „Software Defined Networking" immer weiter in verschiedene Richtungen erweiterte und veränderte. Somit findet sich der Begriff SDN heute auch in Technologien wieder, die mit dem eigentlichen Konzept nur noch wenig zu tun haben. Martin Casado, ein renommierter Wissenschaftler im SDN Umfeld, der auch stark an der Entwicklung der ersten verbreiteteren SDN Technologien, wie OpenFlow oder der Controllerplattform NOX, an der Universität von Stanford beteiligt war, drückte diesen Umstand folgendermaßen aus: „I actually don't really know what SDN means anymore, to be honest. [...] The term was coined in 2009. At that time it meant somthing fairly specific, but now it is just being used as a general term for networking, like all networking is SDN. [...] I think it's an umbrella term for cool stuff in networking" ([1]). Aktuelle Versuche, SDN konkret zu definieren, wie beispielsweise in RFC 7149, beziehen diese Terminologieveränderungen mit ein und fassen SDN viel weiter, als das vorgestellte Konzept von der Trennung von Control und Data Plane.

3. NORTH- UND SOUTHBOUND API

Die Begriffe North- und Southbound Interface existieren auch abseits von SDN. Die Logik hinter der Begriffsbildung ist eingängig: Norden ist oben, Süden ist unten. Hat man nun eine Komponente in einem hierarchisch aufgebauten System, dann ist das Northbound Interface folglich die Schnittstelle dieser Komponente nach oben und somit die Schnittstelle zur in der Hierarchie nächst höheren Instanz, also ein Satz von aufrufbaren Funktionen, die der nächst höheren Instanz von der Komponente präsentiert wird, damit diese die Funktionalität jener nutzen kann. Analog dazu ist das Southbound Interface der Komponente die Schnittstelle nach unten und somit die Schnittstelle zur in der Hierarchie nächst niedrigeren Instanz. Damit ist auch klar, dass North- und Southbound Interface in einem System keine fest definierten Schnittstellen, sondern abhängig von der betrachteten Komponente sind. Konkret auf SDN bezogen, ist diese betrachtete Komponente meist der Controller, wie auch (Abb. 1) veranschaulicht. Das NBI ist folglich die Schnittstelle zwischen dem Controller und den SDN Applikationen. Sie bietet typischerweise eine abstrakte Sicht auf das Netzwerk und Funktionalitäten, mit denen man das Verhalten des Netzes festlegen und Serviceinformationen auslesen kann. Da diese Schnittstelle meist in irgendeiner Form programmierbar ist, spricht man oft anstatt von einem Interface, von einer API (Application Programming Interface). Im Gegensatz dazu bezeichnet man das Southbound Interface nur selten als API, da an der Schnittstelle zwischen dem Controller und den von diesem kontrollierten Netzwerkkomponenten, den Switches, keine Konfigurationsschnittstelle im herkömmlichen Sinne definiert wird. Es handelt sich hier vielmehr um das Protokoll, mit dem der Controller mit den Komponenten kommuniziert, die Menge an von den Switches zu unterstützenden Aktionen, wie zum Beispiel FORWARD, DROP, FLOOD oder Weiterleitung an den Controller, und die Menge an Informationen, die vom Switch bereitgestellt werden müssen. Solche Informationen sind beispielsweise QoS Metainformationen wie Byte- und Packetcounter. Man muss hierbei präzise auf die Terminologie achten. Es gibt auf dem

doi: 10.2313/NET-2014-08-1_13

Markt etliche SDN Lösungen, bei denen verschiedene Controller Frameworks aufeinander aufbauen. Das heißt, dass dort das NBI des hardwarenahen Controllers gleichbedeutend mit dem Southbound Interface des abstrakteren Controllers ist, und die Trennung von Applikation und Controller verschwimmt. Angelehnt daran muss auch beachtet werden, dass in dieser Hierarchie der Controller kein abgeschlossenes, in Software realisiertes, System darstellt, in welchem sämtliche Kontrollinformation, die das Netz steuert, fest implementiert ist. Es muss als Controller Framework betrachtet werden, also als eine Bibliothek, ein Set von Funktionen, die von der Kommunikation am Southbound Interface abstrahiert und Funktionalitäten wie die Verhinderung von Inkonsistenzen bietet. Die eigentlichen Anweisungen, die das gewünschte Verhalten des Netzwerks spezifizieren, kommen in dieser Hierarchie über das NBI des Controllers aus den Applikationen [10] [21].

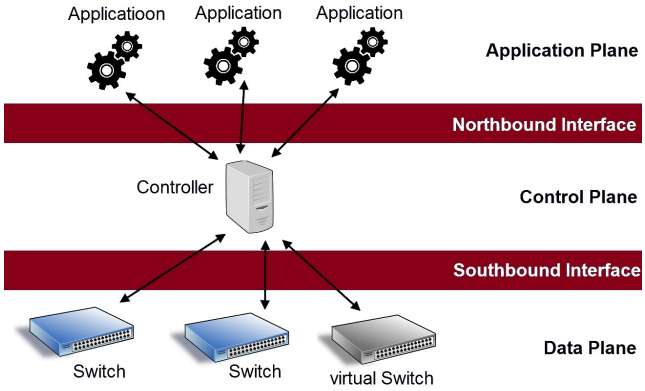


Abbildung 1: Überblick über die SDN Hierarchie

3.1 Southbound Interface OpenFlow

Wie bereits eingangs in der Einleitung beschrieben, wurde das Protokoll OpenFlow von Wissenschaftlern um Nick McKeown von der Universität Stanford zunächst für den Zweck entwickelt, im eigenen Campusnetzwerk Experimente durchführen zu können, wie zum Beispiel ein selbst definiertes Routing Protokoll in einem realen Netz zu testen. Man suchte nach einer Lösung, die mit möglichst wenig Administrationsaufwand sicherstellt, dass der normale Netzbetrieb nicht gestört wird. Das Paper, in dem OpenFlow 2008 vorgestellt wurde, trug dementsprechend auch den passenden Namen „OpenFlow: Enabling innovation in campus networks“. Wie im letzten Absatz festgehalten, definiert OpenFlow als Southbound Interface nicht nur das Kommunikationsprotokoll an sich, das heißt die Abfolge und den Aufbau der Pakete, die zwischen Controller und Switches ausgetauscht werden, sondern auch eine Menge an von den Switches zu unterstützenden Informationen und Aktionen. Bei Einsatz von Hardware Switches ist klar, dass diese das OpenFlow Protokoll explizit unterstützen müssen, weswegen die OpenFlow Entwickler schon von Beginn an eng mit Hardwareherstellern zusammenarbeiten mussten. Das robuste und auf viele Anwendungsgebiete anwendbare und in gewissem Maße auch erweiterbare Design von OpenFlow, in Verbindung mit der stetig ansteigenden Anzahl OpenFlow fähiger Hardware, stellte ein Erfolgskonzept dar und leitete die lawinenartige Verbreitung von OpenFlow basiertem SDN, den „SDN-Hype“, ein.
Einen großen Anteil daran hatte auch die Open Networking Foundation (ONF), eine 2011 gegründete Non-Profit Organisation, welche sich durch eine starke Marktmacht seiner Mitglieder auszeichnet, zu denen unter anderem die Deutsche Telekom, Facebook, Google, Microsoft, Verizon, Yahoo, Netgear, Oracle, HP, Samsung, Intel und IBM gehören. Sie hat sich zum Ziel gesetzt, die Verbreitung und Weiterentwicklung von SDN zu fördern, offene Standards zu schaffen und als erstes Etappenziel OpenFlow als Standard des Southbound Interfaces zu etablieren. Das Design von OpenFlow ist dabei einfach gehalten. Jeder OpenFlow fähige Switch verfügt über eine Pipeline an Flow Tables, in welcher jeder Eintrag einen bestimmten Flow, eine Menge von Paketen mit gleichen Charakteristika, wie beispielsweise gleiche Source- und Destination-IP, repräsentiert. Zu jedem Flow enthält ein Flow Table Eintrag neben den Matchings, welche eben jene Charakteristika spezifizieren, auch eine bestimmte Aktion, die ausgeführt werden soll, wenn ein zum jeweiligen Flow gehörendes Paket am Switch ankommt, und Metainformationen, wie zum Beispiel Timeouts, die Anzahl an bisher empfangenen Bytes oder die Anzahl an bisher empfangenen Paketen. Diese können vom Controller abgefragt oder, im Falle von Informationen wie Timeouts, gesetzt werden. OpenFlow hatte in den letzten Jahren einen großen Einfluss auf die Entwicklung und Weiterentwicklung von SDN. Fast alle namhaften Controller basieren auf OpenFlow, oder unterstützen es zumindest. Und als fester, vom Markt akzeptierter Standard der Schnittstelle am unteren Ende des Controllers ist OpenFlow auch für die folgenden, das NBI behandelnde, Abschnitte von zentraler Bedeutung. Eine ausführlichere Beschreibung von OpenFlow findet sich in den „OpenFlow Switch Specifikations“ der ONF [20]. Ein guter Überblick wird in [3] gegeben.

3.2 Das Northbound Interface

Im Gegensatz zum Southbound Interface hat sich am Northbound Interface bisher noch keine Schnittstelle als Standard durchsetzen können. Weshalb dies so ist und welche Folgen sich daraus ergeben, soll im nachfolgenden Abschnitt erläutert werden. Da diese Überlegungen sehr aktuell sind, basieren viele der folgenden Informationen nicht auf etablierten wissenschaftlichen Arbeiten, sondern sind häufig aus Blogs, Forenbeiträgen und Internetartikeln erfahrener und kompetenter Wissenschaftler und Forscher entnommen. Wichtige Anlaufstellen sind dabei die Websites „www.sdncentral.com“ und „searchsdn.techtarget.com“. Wichtige Artikel, aus welchen ein Großteil der in Abschnitt 3.2 und 4 vorgestellten Informationen entnommen wurde, sind [24], [9], [16], [12], [5] und [6]. Es sollte dem Leser deshalb bewusst sein, dass einerseits, auch wenn in dieser Arbeit versucht wird die mit Stand Q2/2014 aktuell allgemein vorherrschende Meinung korrekt wiederzugeben, der Inhalt dennoch wesentlich von der Sichtweise dieser Experten abhängt, und dass andererseits zu vielen der untersuchten Themen aktuell noch kein wissenschaftlicher Konsens gefunden ist.
Die aktuelle Situation gestaltet sich wie folgt: Es befinden sich derzeit mehr als 30 gängige Controller Frameworks auf dem Markt [25]. Alle diese Controller wurden von verschiedensten Instanzen mit jeweils sehr individuellen Zielen zur Realisierung eines ganz spezifischen Zwecks geschaffen. Es gibt proprietäre Controller, Controller von Open Source Ent-

doi: 10.2313/NET-2014-08-1_13

wicklern, Controller die an Universitäten geschaffen wurden, einige bieten die Basis für eine Palette von kommerziellen Netzwerkprodukten, andere sind eher von experimenteller Natur, einige sind ausgefeilt, bieten großen Funktionsumfang oder sind auf möglichst hohe Effizienz oder Ressourcensparsamkeit getrimmt, während andere nur spezifische Aufgaben erfüllen oder mit dem primären Ziel geschaffen wurden, möglichst einfach zu bedienen zu sein. Dementsprechend arbeiten diese Controllerplattformen auch auf völlig verschiedenen Abstraktionslevels, sowohl bezüglich Abstraktion von der Topologie, als auch im Hinblick auf den Abstraktionsgrad der Konfigurationsschnittstelle. Zudem sind die Schnittstellen der Controller in die verschiedensten Programmiersprachen eingebettet. Folglich sind aktuell mehr als 30 verschiedene NBIs im Umlauf. Diese Situation kann zu dem Schluss führen, dass die in Kapitel 2 angesprochene Portabilitätsproblematik, die das herstellerübergreifende Übertragen von Konfiguration erschwert und von SDN in Verbindung mit OpenFlow als standardisiertem Southbound Interface eigentlich beseitigt werden sollte, von den Switches hin zu den Controllern verlagert wurde. An dieser Stelle setzen nun die Überlegungen bezüglich der Schaffung eines Standards für das NBI an. Denn wie OpenFlow am Southbound Interface, würde ein Standard diese Probleme beseitigen [9][19][5].

4. NBI STANDARISIERUNG

Wie bereits angedeutet ist aktuell noch kein konkreter Standard in Sicht. Es gibt aber durchaus Projekte, aus denen ein Standard hervorgehen könnte. Darüber hinaus wird in Fachkreisen eine lebhafte Diskussion über die Vorteile und Probleme einer Standardisierung, sowie über die Anforderungen an NBI und NBI Standard geführt. Diese Vorgänge sollen in diesem Abschnitt zusammengefasst werden.

4.1 Aktueller Stand der Standardisierung

Erfolgsversprechende Bestrebungen gehen aktuell von der ONF aus. Diese hatte bereits 2012 eine Diskussionsgruppe zum Thema NBI Standard ins Leben gerufen. Man kam zu dem Schluss, dass der Zeitpunkt für eine Standardisierung noch nicht reif sei, und beschloss zunächst nur Use-Cases zu erfassen und diese Arbeit in eine andere Arbeitsgruppe, die „Architecture and Framework Working Group“, einzugliedern. Aufgrund des immer stärker werdenden Drucks, den einige der Mitglieder innerhalb der ONF ausgeübt haben, wurde nun im Oktober 2013 auch eine offizielle „Northbound Interfaces Working Group“ gegründet [5][8]. Die Leitung wurde an Sarwar Raza, Direktor von Cloud-Networking und SDN in der Advanced Technology Group for Networking bei Hewlett-Packard, vergeben. Die Arbeitsgruppe hat bis jetzt außer einem Charter [19], in welchem kurz ihre Ziele, Aufgaben und Vorgehensweisen beschrieben werden, zwar noch keine Ergebnisse veröffentlicht, dies ist bei der ONF aber auch nicht untypisch, da diese nur selten Resultate präsentiert solange der Arbeitsprozess noch nicht abgeschlossen ist. In dem zu Anfang veröffentlichten Charter wird klar festgehalten, dass sich die NBI Working Group nicht das Ziel gesetzt hat, zwingend einen Standard zu etablieren. Vielmehr soll festgestellt werden, inwiefern Standardisierung sinnvoll, beziehungsweise durchführbar ist, also ähnliche Überlegungen angestellt werden, wie sie in den folgenden Abschnitten beschrieben werden. Die ONF verfolgt dabei einen auf der Evaluation von Use-Cases aufbauenden Ansatz. Es wird in dem Charter ebenfalls betont, dass, falls die NBI Working Group schließlich doch zu der Entscheidung finden sollte, selbst einen Standard etablieren zu wollen, vor konkreten Schritten in Richtung Standardisierung unbedingt zuerst eine lauffähige Implementierung des zu standardisierenden Konzepts geschaffen und in den Markt eingeführt werden muss. Die ONF will dabei nicht nur mit Entwicklungsabteilungen ihrer Mitglieder, sondern unter Umständen auch mit externen Entwicklern, auch mit Entwicklern aus Open Source Projekten, zusammenarbeiten. Die Bestrebungen der ONF werden auch in einigen Artikeln, wie [5], [8], [13] und [14] beschrieben und bewertet.

Neben der ONF bemüht sich derzeit keine andere Institution, wie beispielsweise IEEE oder IETF, um einen Standard am NBI. Sollte die ONF jedoch zu keinen Ergebnissen kommen und der Wunsch nach einem Standard vonseiten des Marktes weiter steigen, ist es auch nicht ausgeschlossen, dass sich diese Institutionen zukünftig intensiver auf diesem Feld betätigen werden [8]. Neben den Bemühungen der ONF sehen einige Experten das OpenDaylight Projekt [26] als erfolgversprechendsten Ausgangspunkt für eine Standardisierung [5]. OpenDaylight ist ein Open Source Projekt, welches von der Linux Foundation verwaltet wird und auf der Unterstützung namhafter Unternehmen im Netzwerksektor, wie den Platin Mitgliedern Cisco, HP, IBM, Jupiter Networks, Redhat und Microsoft, aufbaut. Es wurde im April 2013 ins Leben gerufen und hat zum Ziel, ein breit gefächertes Framework für Anwendungen im Bereich des Software Defined Networkings und des Network Functions Virtualizations zu erschaffen. Erster Code wurde im Februar 2014 unter dem Namen Hydrogen veröffentlicht. OpenDaylight baut auf einer eigenen Java Virtual Machine (JVM) auf, bietet Unterstützung für das auf Java basierende OSGi-Framework, welches die Verwaltung von Softwarekomponenten und Modularisierung vereinfacht, und nutzt das Konzept der RESTful APIs [26]. Dieses Konzept findet in vielen modernen API Konzeptionen Anwendung. Es basiert auf der Dissertation von Roy Fielding [4] und definiert 6 Richtlinien für Representational State Transfer. Eine genaue Einführung in das Innenleben von OpenDaylight oder eine genaue Beschreibung der dort verwendeten APIs übersteigt den Rahmen dieser Arbeit bei Weitem, und ist auch nicht sonderlich sinnvoll, da Hydrogen den ersten Release repräsentiert, und sich das Framework daher in naher Zukunft stark verändern wird. Das OpenDaylight Projekt ist noch jung und hauptsächlich an der Produktion von Code, nicht am Etablieren eines Standards, interessiert. Daher ist eine offizielle Standardisierung vonseiten OpenDaylights trotz der Marktmacht seiner Unterstützer unwahrscheinlich. Es ist aber durchaus möglich, dass es sich in den kommenden Jahren auf dem Markt als De-facto-Standard durchsetzt [5].

4.2 Vorteile einer Standardisierung

In Abschnitt 3.2 wurden bereits Probleme angesprochen, die von einem NBI Standard gelöst werden könnten. Um die Bedeutung der Diskussion um die Standardisierung, beziehungsweise den wachsenden Wunsch nach einem Standard, vollständig verstehen zu können, ist es nötig, sich über sämtliche Vorteile einer Standardisierung klar zu werden. Ein schon behandelter Vorteil ist die Verbesserung der Nutzerfreundlichkeit durch Steigerung der Portabilität. Baut eine Anwendung auf einer standardisierten Schnittstelle auf, ist es möglich diese auf verschiedenen Controllern, die die-

doi: 10.2313/NET-2014-08-1_13

se standardisierte Schnittstelle nutzen, zu übertragen. Dies kann für die Nutzer einen enormen Mehrwert darstellen, denkt man beispielsweise wieder an einen Wechsel der Controllerplattform, den Aufbau von Multi-Vendor Umgebungen oder die Entwicklung einer Applikation, die auf verschiedenen Controllern ordnungsgemäß funktionieren sollen. Letzteres ist eine sehr zentrale Anforderung, da es sich bei den Nutzern des NBIs auch um die Entwickler von Applikationen handelt. Möchte man eine Applikation entwickeln die größere Teile des Marktes erreicht, muss diese auf verschiedenen, konkurrierenden Controllern lauffähig sein, und dazu ist es derzeit unumgänglich das eigene Programm für jeden einzelnen Controller, beziehungsweise dessen NBI umzuschreiben. Dies stellt auch eine Verschwendung von Ressourcen dar, da Ressourcen, die derzeit für die Portierung der Applikationen auf verschiedene Systeme und die Pflege der Programme auf den verschiedenen Systemen, bei Existenz eines Standards sinnvolleren Verwendungen, wie beispielsweise der Steigerung der Performance oder der Usability zugunsten kommen könnten. Sowohl auf standardisierten Schnittstellen aufbauende Applikationen, als auch standardisierte Schnittstellen unterstützende Controller erreichen folglich mehr Endnutzer. Dies bringt weitere Vorteile, wie zum Beispiel, dass auch vergleichsweise kleine Unternehmen größere Chancen auf dem SDN Markt haben. Dadurch, und dadurch, dass verschiedene Controllerhersteller nun nicht mehr hauptsächlich über die Gestaltung der Northbound API, beziehungsweise Anwendungsentwickler nicht mehr hauptsächlich über den zugrundeliegenden Controller konkurrieren, sondern Faktoren wie Performance, Stabilität und Sicherheit sehr viel größere Bedeutung erhalten würden, könnte die veränderte Konkurrenzsituation den Markt auch stark beleben. Man sollte an dieser Stelle auch bedenken, dass das Wichtigste im SDN Umfeld die Applikationen sind. Denn auch wenn die Controller und die Netzwerkgeräte auf der Data Plane wichtige Bestandteile der SDN Hierarchie sind, stellen sie letztendlich nur die Werkzeuge dar, um auf dem Netzwerk ausgefeilte und mächtige Applikationen realisieren zu können. Damit sind Bestrebungen, die die Anwendungsentwickler in ihrer Arbeit unterstützen, besonders wichtig und, da eine Standardisierung des NBIs natürlich insbesondere den Anwendungsentwickler zugute kommen würde, gilt es den Wert der vorgestellten Vorteile besonders stark zu gewichten [12][6][19].

4.3 Anforderungen an ein NBI

Um erfolgreich einen neuen Controller und dessen NBI entwickeln zu können, muss der Entwickler zu Beginn genau spezifizieren, welche Anforderungen er an jene stellen möchte. Dies ist selbstverständlich auch bei der Entwicklung eines standardisierten NBIs, beziehungsweise eines NBI Standards, nötig. Hier gestaltet sich diese Aufgabe jedoch deutlich komplexer, da ein Standard möglichst viele der unterschiedlichen und sich teilweise auch widersprechenden Anforderungen abdecken soll: Eine sinnvolle Anforderung ist beispielsweise ein für den Anwendungsentwickler angemessener Funktionsumfang, angemessener Abstraktionsgrad und angemessene Komplexität. Doch wer ist der Entwickler und was ist für ihn angemessen? Entwickler bei Internet Service Providern oder in Rechenzentren werden meist nicht durch hohe Komplexität beeinträchtigt. Sie sind unter Umständen sogar auf einen großen Funktionsumfang und ein niedriges Abstraktionslevel angewiesen. Netzwerkadministratoren

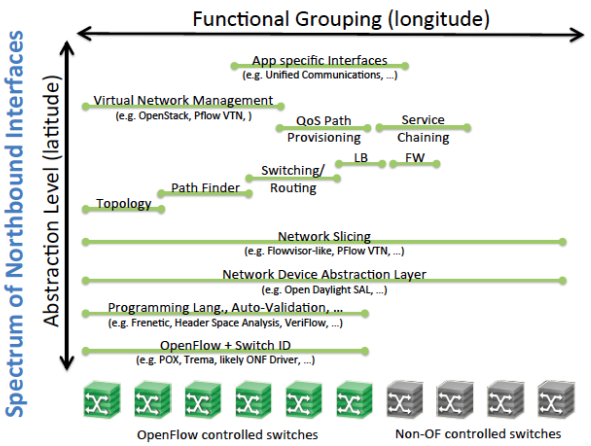


Abbildung 2: Abstraktionslevel am Northbound Interface, entnommen aus [19]

oder kleine Unternehmen, die SDN Apps entwickeln wollen, deren Schwerpunkt möglicherweise gar nicht auf SDN liegt, hingegen benötigen dies nur selten und favorisieren geringe Komplexität und einen etwas höheren Grad an Abstraktion bezüglich der eigentlichen Kommunikation zwischen Controller und Switches.

Eine Übersicht über verschiedene Ebenen, auf welchen SDN-Anwendungsentwickler operieren können, findet sich in (Abb. 2), einer Abbildung aus dem initialen Charter der NBI Working Group [19]. Wie dargestellt, arbeiten Entwickler von Virtual Network Management Applikationen auf einem viel höheren Level als Low-Level Controller wie POX. Somit erweist sich ein Low-Level NBI wie das von POX für diese Entwickler häufig als ungeeignet. Auch bei der Abstraktion von der tatsächlichen physischen Topologie verhält sich dies ähnlich. Entwickler, die Anwendungen entwickeln, die von der realen Topologie weitgehend unabhängig sind, wollen häufig auch nur möglichst wenig von dieser wahrnehmen, um beispielsweise eine einfache Wiederverwendbarkeit der Module gewährleisten zu können. Entwickler in Rechenzentren hingegen, die ihre Topologie genau kennen und wissen, dass sich diese nur selten verändert, möchten häufig dieses Wissen auch ausnutzen, um zum Beispiel möglichst effizientes Load Balancing betreiben zu können. Einige Entwickler bevorzugen bestimmte Lizenzmodelle, unter welchen Controller und damit auch NBI veröffentlicht werden. Weitere Anforderungen an eine Northbound API können aus Anforderungen hervorgehen, die man an den zugrundeliegenden Controller stellt. Dazu zählt beispielsweise gute Skalierbarkeit oder die Möglichkeit des Betriebs eines verteilten Controllers. Auch solche Anforderungen haben Einfluss auf die Gestaltung der jeweiligen Northbound API und können ebenfalls mit anderen Anforderungen, wie einem möglichst großen Funktionsumfang oder möglichst geringer Komplexität in Konflikt stehen.

Letztendlich lässt sich also festhalten, dass ein abgeschlossener, wohl definierter Satz von Anforderungen an das NBI, der alle Wünsche sämtlicher Entwickler abdeckt, nicht existieren kann. Stattdessen hängen diese immer vom individuellen Verwendungszweck ab, was bereits die Kernproblematik des nächsten Abschnitts darstellt [14][19][6].

doi: 10.2313/NET-2014-08-1_13

4.4 Probleme einer Standardisierung

In Abschnitt 4.2 wurden die Vorteile vorgestellt, die eine Standardisierung mit sich bringen würde. Dennoch existiert wie in Abschnitt 4.1 dargelegt bisher noch kein Standard. Dies liegt zunächst einmal an der Vielzahl an verschiedenen Akteuren in Forschung, Entwicklung und auf dem freien Markt, wovon jeder unterschiedlichste Interessen, Vorstellungen und Visionen über die (optimale) Zukunft von SDN vertritt. Jeder dieser Akteure verwendet SDN für seine individuellen Anwendungszwecke, was wie in Abschnitt 4.3 beschrieben dazu führt, dass jeder ein individuelles Set an Anforderungen mitbringt und daraus die Frage entsteht, welche dieser konkurrierenden Anforderungen der Standard verwirklichen soll. Durch die Vielzahl und Vielfalt an Anforderungen gehen die meisten Experten davon aus, dass eine einzige API nicht ausreichen wird und ein möglicher Standard mehrere APIs standardisieren müsste [5][11]. Ein anderer Ansatz wäre, die Schnittstellen stark erweiterbar zu gestalten. Daraus ergibt sich die Fragestellung wie granular man den Standard aufbauen sollte und die Gefahr, dass bei zu großer Erweiterbarkeit, beziehungsweise der zu starken Fragmentierung des Standards in verschiedene APIs, nach einigen Jahren der Verbesserungen, Weiterentwicklung, Anpassung des Standards an neue Anforderungen und dem daraus resultierenden Aufkommen von nicht kompatiblen Versionen, die Situation am NBI wieder genauso komplex, facettenreich und mit Redundanz versehen sein wird, wie sie zum jetzigen Zeitpunkt schon ist. Ein weiteres Problem sind Akteure, die nicht am Erfolg von auf offenen Standards basierendem SDN interessiert sind, sondern es vorziehen würden, ihre Marktmacht ausnutzen zu können, um mit ihren proprietären Lösungen den Markt zu beherrschen. Generell ist der von Software geprägte Markt am NBI sehr dynamisch und verändert sich mitsamt den gängigen Anforderungen relativ schnell, was an sich schon im Widerspruch zu den meist langwierigen Standardisierungsprozessen steht. Standardisierung von hardwareunabhängigen Softwarekomponenten gestaltet sich meist schwierig. Aufgrund all dieser Probleme stehen einige Akteure einer Standardisierung generell skeptisch gegenüber, sodass die Gefahr besteht, dass jene einen veröffentlichten Standard einfach nicht annehmen. Hieraus folgt auch bereits das größte Hindernis, welches ein möglicher Standard überwinden müsste: Mangelnde Marktakzeptanz. Software-Standards sind bei Weitem nicht so bindend wie Hardware-Standards, und damit ist Marktakzeptanz bei Standardisierung von Softwareschnittstellen viel bedeutender. Ein Standard den niemand benutzt ist nutzlos und im eigentlichen Sinn auch kein Standard [6][13][14].

4.5 Anforderungen an einen NBI Standard

Aus den Hindernissen, die der Standardisierung des NBIs im Wege stehen, lassen sich schließlich auch die Anforderungen an einen NBI Standard ableiten. Man sollte zunächst bei der Gestaltung des Standards vorsichtig mit Inkompatibilität verursachender Erweiterbarkeit umgehen, und stets versuchen, bei der Weiterentwicklung des Standards die entstehenden Versionen kompatibel zu halten. Redundanz ist ebenfalls schädlich und gefährdet jede Form von Standard. Deshalb muss besonders bei der Weiterentwicklung die Einführung redundanter Komponenten in den Standard verhindert werden. Bei Use-Case basierten Ansätzen müssen zukünftige Veränderungen der Use-Cases schon im Vorfeld mit eingeplant und bei der Weiterentwicklung auch neue, gängige Use-Cases mit aufgenommen werden. Dabei sollten nur die wichtigsten Use-Cases miteinbezogen werden. Ein Standard wird nicht sämtliche Anwendungsgebiete bis ins Detail abdecken können, weshalb es wichtig ist die Grenzen des Standards von Anfang an klar abzustecken. Auch hier hängen die konkreten Anforderungen an den Standard wieder von den anvisierten Verwendungszwecken ab [14]. Schließlich sollte ein Standard nach gängiger Meinung (z.B. in [19]) auch unabhängig von konkreten Programmiersprachen gestaltet werden. Auch die ONF will wenn dann nur ein abstraktes Datenmodell standardisieren. Konkrete Implementierungen soll der Markt liefern. Eine Referenzimplementierung sollte jedoch schon vor der eigentlichen Standardisierung in den Markt eingeführt werden, da sich nur so die wichtigste Anforderung an einen Standard erfüllen lässt: Marktakzeptanz erlangen [19][14].

4.6 Unterschied im Standardisierungsprozess von North- und Southbound Interface

Es wurde nun in Abschnitt 3.1 das Southbound Interface OpenFlow und dessen Entwicklung zum Standard behandelt und in Abschnitt 4 ausführlich auf das NBI und dessen Standardisierung eingegangen, sodass nun rekapituliert und die Unterschiede im Standardisierungsprozess von North- und Southbound Interface diskutiert werden kann. Das Southbound Interface definiert Anforderungen an Hardware, Hardwareentwicklungszyklen sind lang und Hardware Entwicklung daher sehr kostspielig. Da diese großen Investitionen meist nur von den großen, traditionellen Netzwerkgeräteherstellern getätigt werden können, haben diese bei Entwicklung am Southbound Interface einen großen Vorteil. Kleine Unternehmer und Newcomer sind benachteiligt. Der Markt ist deshalb verhältnismäßig statisch und beherbergt nur wenige Akteure, weswegen sich Standardisierungsprozesse dementsprechend unproblematisch gestalten. Das NBI hingegen ist eine Softwareschnittstelle. Hier können sich kleine Hersteller einfacher und schneller etablieren und es sind dadurch mehr Akteure auf dem Markt tätig. Er ist diversifizierter, dynamischer und regelt vieles selbst. Die Standardisierung des NBIs in diesem dynamischen Umfeld ist, wie bei Standardisierung von Softwareschnittstellen häufig zu beobachten, sehr schwierig [6][11].

5. NORTHBOUND API PYRETIC

Abschließend soll in dieser Arbeit ein konkretes NBI beschrieben werden. Die Wahl fiel hierbei auf die Open Source Plattform Pyretic. Denn Pyretic bietet eine übersichtliche, einfach zu verstehende und relativ abstrakte Northbound API, was einen starken Kontrast zu den NBIs einiger häufig benutzter Controllerplattformen darstellt, wie beispielsweise der verbreiteten Plattform NOX, welche im engeren Sinn nur Funktionen liefert, die als Wrapper für die OpenFlow Kontrollnachrichten betrachtet werden können. Pyretic wurde vor allem an den Universitäten Cornell und Stanford entwickelt und 2013 in [17], beziehungsweise [23], vorgestellt. Pyretic gehört zu der bereits 2010 vorgestellten Frenetic Familie, verfügt im Gegensatz zu Frenetic allerdings über sequentielle Komposition (siehe Abschnitt 5.2), ist anstelle von Ocaml in die Programmiersprache Python eingebettet und basiert in der aktuellen Implementierung auf dem Controller POX, statt wie die Implementierung von Frenetic auf dem Controller NOX. Durch die von Pyretic gebotene Möglich-

doi: 10.2313/NET-2014-08-1_13

keit in wenigen, leicht verständlichen Zeilen Code mächtige Applikationen zu schreiben, in Zusammenspiel mit der Pyretic Laufzeit, welche inkonsistente Konfiguration der OpenFlow Switches weitgehendst verhindert, wird das Risiko von ungewolltem oder unvorhersagbarem Netzwerkverhalten minimiert und schnelles Prototyping vereinfacht. Ein Lizenzmodell im Stil der Berkeley Software Distribution (BSD) Lizenz erfüllt dabei die Anforderungen von kommerziellen, wie auch von nicht kommerziellen Entwicklern (vgl. [23], S. 1). Einen Blick auf Pyretics NBI zu werfen ist deshalb durchaus sinnvoll, auch wenn Pyretic an sich, wie in Abschnitt 4.4 beschrieben, keinen großen Stellenwert auf dem SDN Markt inne hat.

5.1 Das abstrakte Paket Modell

Aus Sicht Pyretics sind Pakete Wörterbücher, die bestimmte Felder, wie Header Informationen, Paketdaten, Standort, also das Tupel aus der Identität eines virtuellen oder physischen Gerätes und dem jeweiligen Port, aber auch virtuelle, selbst definierte Headerfelder, auf ihre tatsächlichen Werte, beziehungsweise einen Stack von Werten, mappen. Dieser Stack wird dazu benutzt, verschiedene Ebenen der Abstraktion zu realisieren. So kann zum Beispiel der Standort Wert eines Paketes ein Stack sein, der auf der untersten Ebene die Identität des physischen Switches, in dem sich das Paket gerade befindet, und in verschiedenen Ebenen darüber die Identitäten der virtuellen Switches der virtuellen Netze, in dem sich das Paket logisch aufhält, enthalten. Auf diesen Grundkonzepten aufbauend bietet Pyretic ein ausgefeiltes Konzept zur Topologieabstraktion. Dieses basiert auf „Network Objects“, die aus einer Topologie, einer Policy und einem Mapping, welches die Elemente der Topologie mit denen der darunterliegenden Topologie verknüpft, bestehen. Pyretic bietet dabei Funktionalitäten, die es ermöglichen automatisch Mappings zu generieren um Veränderungen einer Topologie auf andere Topologieabstraktionsebenen abzubilden oder semantisch gleiche Policies auf verschiedene Topologien zu übertragen. Mithilfe dieser Network Objects lassen sich auch die Konzepte „information hiding“ und „information protection“, also Zugriffssteuerung auf Informationen, umsetzen [17].

5.2 Modularisierung und Komposition

In modernen SDN-Applikationen ist besonders im industriellen Umfeld und bei Verwendung der gängigen „Low Level NBI Controllern“ hohe Komplexität ein großes Problem. Pyretic versucht sich diesem Problem anzunehmen, indem es dem Entwickler erlaubt seinen Code sehr modular zu gestalten und sämtliche Funktionalität in Modulen zu kapseln. Solche Module lassen sich dann einfach durch zwei verschiedene Kompositionsoperatoren verknüpfen. Die erste Möglichkeit ist die Verknüpfung zweier Module mittels paralleler Komposition, ausgedrückt durch den + Operator. Hierbei werden die Module unabhängig voneinander parallel ausgeführt, was ungewollte Interferenzen zwischen den Funktionen und damit Inkonsistenzen verhindert. Die zweite Möglichkeit ist die sequentielle Komposition, welche durch den << Operator dargestellt wird. Diese sorgt dafür, dass Pakete, die vom ersten Modul zurückgegeben werden dem zweiten Modul als Eingabe dienen. Dieses Modell ermöglicht starke Modularisierung des Codes, wodurch die Portabilität und Wiederverwendbarkeit von Codesegmenten gesteigert, und damit auch Redundanz im Code minimiert werden kann.

5.3 High-Level Policy Funktionen

Die Bewegung und Verarbeitung von Paketen wird in Pyretic durch Funktionen dargestellt, die ein Paket als Eingabe erhalten und eine Menge von Paketen zurückgeben. Diese Menge kann sowohl leer sein, wie zum Beispiel im Falle einer DROP Policy, oder auch ein oder mehrere verschiedene Pakete beinhalten. Die Werte der Felder der abstrakten Paketdarstellung können bei diesen Paketen je nach Definition der Funktion beliebig verändert werden. So wird eine einfache, statische Weiterleitung eines Paketes, welche alle Pakete die an Port A eines Netzwerkgerätes X ankommen unverändert auf Port B wieder ausgeben soll, durch eine Funktion umgesetzt, die bei einem eingehenden Paket mit den Standort Werten „Gerät X, Port A“ ein einzelnes Paket zurückgibt, bei dem der Wert des Standort Feldes, spezifischer des Port Feldes, auf B gesetzt ist und die restlichen Werte vom eingehenden Paket übernommen wurden. Auf diesem Modell aufbauend werden verschiedene einfache High-Policy Funktionen definiert, die als NetCore bezeichnet werden: Es gibt zunächst einige leicht zugängliche Primitive:

A ::= drop | passthrough | fwd(port) | flood | push(h=v) | pop(h) | move(h1=h2)

Diese werden durch Prädikate ergänzt, die sich durch einfache boolesche Operatoren verknüpfen lassen:

P ::= all_packets | no_packets | match(h=v) | ingress | egress | P & P | (P | P) | ∼ P

Damit der Controller Informationen von den Switches abfragen kann, werden Query Policies eingeführt. Diese übergeben die angeforderten Informationen an „Bucket“ genannte Datenstrukturen, auf welche „Listeners“ wie Callback-Funktionen registriert werden können. *packets(n,[h])* übergibt die jeweils ersten n Pakete, die gleiche Werte in den in der Liste *[h]* bestimmten abstrakten Paketfeldern vorweisen, vollständig und unverändert an das Bucket *packet_bucket*. *counts(every,[h])* hingegen ruft alle *every* Sekunden alle registrierten Listener auf und übergibt ihnen über das Bucket *counting_bucket* eine Abbildung von Paketen der gleichen in *[h]* definierten Eigenschaften auf die Anzahl an Paketen, auf die diese Eigenschaften zutreffen.

Q ::= packets(limit,[h]) | counts(every,[h])

Diese einfachen Policies lassen sich dann mittels der bereits vorgestellten sequentiellen und parallelen Komposition verknüpfen, sich durch Anwendung eines Prädikates oder einer if-else Konstruktion konditionieren und sich somit zu komplexeren statischen Policies zusammenfügen.

C ::= A | Q | P[C] | (C | C) | C >> C | if_(P,C,C)

Des Weiteren kann eine Policy ihr eigenes Verhalten selbst modifizieren, wodurch die Möglichkeit geschaffen wird, dynamische, zur Laufzeit veränderliche Policies zu erzeugen. Ausführliche Beispiele und Erklärungen zu allen vorgestellten Inhalten finden sich in [17], [23] und [2].

5.4 Grenzen von Pyretic

Wie bereits erwähnt, bietet Pyretic selbst unerfahrenen Entwicklern die Möglichkeit in kurzer Zeit mit wenig Code mächtige Applikationen zu schreiben. Dennoch soll hier nicht der Eindruck erweckt werden, dass Pyretic revolutionär neue

doi: 10.2313/NET-2014-08-1_13

Ideen umsetzt oder gar einen großen Stellenwert auf dem Markt einnimmt. Pyretic ist ein gutes Beispiel für eine einfache NBI-Sprache, die schöne Konzepte im Hinblick auf Abstraktion bietet und Lösungen für aktuelle NBI-Probleme sucht, aber dennoch klar gesetzten Grenzen unterworfen ist: Pyretic ist, auch durch die Verwendung von POX als Basis, einzig auf OpenFlow ausgerichtet. Unterstützung von anderen Southbound Interfaces ist nicht vorgesehen. Pyretic wurde bisher in keiner veröffentlichten Publikation evaluiert, weder im Hinblick auf Korrektheit, noch im Hinblick auf Usability, Stabilität, Skalierbarkeit oder der besonders wichtigen Performance. Da Pyretic auf POX basiert und dieser in Performanceuntersuchungen sehr schlecht abschneidet [25] und daher für realitätsnahe industrielle Entwicklung als unbrauchbar klassifiziert wird, ist davon auszugehen, dass auch Pyretic, zumindest in der aktuellen Implementierung, keine ausreichend gute Performance bietet, um sich über das Forschungsumfeld hinaus etablieren zu können [17]. Pyretic ist noch jung, weswegen bisher nur wenige Applikationen in Pyretic realisiert wurden. Und letztendlich ist Pyretic auch vergleichsweise stark an Python gebunden, was eine Portierung in andere Sprachen schwierig machen kann. Aufgrund dieser Umstände wird Pyretic selbst kaum große Bedeutung gewinnen, es liefert allerdings Ansätze für neu entwickelte Sprachen und wertvolle Anregungen für einen künftigen Standard.

6. FAZIT

Es wurde in dieser Arbeit das Konzept, die Anwendungsgebiete, der State of the Art und die Bedeutung von SDN erläutert. Es wurde der Einfluss und die Rolle von OpenFlow als Southbound Interface behandelt und detailliert das NBI aus unterschiedlichen Blickwinkeln betrachtet. Im letzten Kapitel wurden einige der etwas abstrakteren Überlegungen anhand des Überblicks über eine konkrete Northbound API veranschaulicht. Aufgrund dieser Ausführungen wurde die derzeit gegebene Situation am NBI, die Probleme, die es aktuell zu bewältigen gilt, und die Fähigkeiten und das Wissen, moderne Trends und Entwicklungen am NBI zumindest grundlegend bewerten zu können, in ausreichendem Maße vermittelt. Dem Leser sollte nun bewusst sein, wie stark die Entwicklungen am NBI sämtliche Bereiche des Software Defined Networkings prägen, und welch großen Einfluss diese Entwicklung dadurch auch auf den gesamten Netzwerksektor hat. Somit bleibt an dieser Stelle nur noch festzuhalten, dass das NBI nicht nur für SDN Entwickler von Interesse ist, sondern in der Tat eine innovative Internettechnologie darstellt, also den Grundgedanken des Seminars, in dessen Rahmen diese Ausarbeitung verfasst wurde, in jeder Beziehung trifft, und daher für alle, die sich mit modernen Netzwerktechnologien auseinandersetzen oder einfach nur aktuelle Innovationen in der Informatik mitverfolgen wollen, ein attraktives Themengebiet verkörpert, und die Empfehlung auszusprechen, die Entwicklung, die das Thema Northbound Interface in den nächsten Monaten und Jahren erfahren wird, ein bisschen im Auge zu behalten.

7. LITERATUR

[1] OpenFlow Inventor Martin Casado on SDN, VMware, and Software Defined Networking Hype. `http://www.enterprisenetworkingplanet.com/netsp/openflow-inventor-martin-casado-sdn-vmware-software-defined-networking-video.html`, aufgerufen am 07.06.2014.

[2] Pyretic Homepage. `http://frenetic-lang.org/pyretic/`, aufgerufen am 07.06.2014.

[3] Z. Bozakov. OpenFlow eine Softwareschnittstelle zur Netzprogrammierung. *ix Magazin für professionelle Informationstechnik*, 8(9):112–115, Juli 2012.

[4] R. T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, 2000.

[5] R. Gewirtz. ONF to standardize northbound API for SDN applications? `http://searchsdn.techtarget.com/news/2240206604/ONF-to-standardize-northbound-API-for-SDN-applications`, aufgerufen am 07.06.2014, Oktober 2013.

[6] I. Guis. The SDN Gold Rush To The Northbound API. `http://www.sdncentral.com/technology/the-sdn-gold-rush-to-the-northbound-api/2012/11/`, aufgerufen am 07.06.2014, November 2012.

[7] U. Holzle. OpenFlow @ Google. `http://www.opennetsummit.org/archives/apr12/hoelzle-tue-openflow.pdf`, aufgerufen am 07.06.2014.

[8] S. Johnson. Do SDN northbound APIs need standards? `http://searchnetworking.techtarget.com/feature/Do-SDN-northbound-APIs-need-standards`, aufgerufen am 07.06.2014.

[9] S. Johnson. A primer on northbound APIs Their role in a software-defined network. `http://searchsdn.techtarget.com/feature/A-primer-on-northbound-APIs-Their-role-in-a-software-defined-network`, aufgerufen am 07.06.2014, Dezember 2012.

[10] K. Kirkpatrick. Software-defined networking. *Communications of the ACM*, 56(9):16–19, September 2013.

[11] D. Kreutz, F. M. V. Ramos, P. Veríssimo, C. E. Rothenberg, S. Azodolmolky und S. Uhlig. Software-defined networking: A comprehensive survey. *CoRR*, Juni 2014.

[12] D. Lenrow. The Most Important Work in SDN: Have We Got It Backward? `http://www.sdncentral.com/education/important-work-sdn-got-backwards/2014/04/`, aufgerufen am 07.06.2014, April 2014.

[13] S. McGillicuddy. ONF launches SDN northbound interface working group. `http://searchsdn.techtarget.com/news/2240207495/ONF-launches-SDN-northbound-interface-working-group`, aufgerufen am 07.06.2014, Oktober 2013.

[14] S. McGillicuddy. ONF Northbound Interface Group chair: No need for standards yet. `http://searchsdn.techtarget.com/news/2240210605/ONF-Northbound-Interface-Group-chair-No-need-for-standards-yet`, aufgerufen am 07.06.2014, Dezember 2013.

[15] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker und J. Turner. OpenFlow: Enabling Innovation in Campus

Networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, März 2008.

[16] J. Metzler. Where Do We Stand with SDN's Northbound Interface - Webtorials. http://www.webtorials.com/content/2014/04/where-do-we-stand-with-sdns-northbound-interface.html, aufgerufen am 07.06.2014, April 2014.

[17] C. Monsanto, J. Reich, N. Foster, J. Rexford und D. Walker. Composing Software-defined Networks. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, nsdi'13, Seiten 1–14, Berkeley, CA, USA, 2013. USENIX Association.

[18] Open Networking Foundation. Software-Defined Networking: The New Norm for Networks. https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf, aufgerufen am 07.06.2014, April 2012.

[19] Open Networking Foundation. Northbound Interfaces. https://www.opennetworking.org/images/stories/downloads/working-groups/charter-nbi.pdf, aufgerufen am 07.06.2014, Oktober 2013.

[20] Open Networking Foundation. OpenFlow Switch Specification Version 1.4.0 (Wire Protocol 0x05). https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf, aufgerufen am 07.06.2014, Oktober 2013.

[21] Open Networking Foundation. SDN Architecture Overview. https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/SDN-architecture-overview-1.0.pdf, aufgerufen am 07.06.2014, Dezember 2013.

[22] B. M. Posey. What You Need to Know About Software Defined Networking in Hyper-V. http://www.virtualizationadmin.com/articles-tutorials/microsoft-hyper-v-articles/networking/what-you-need-know-about-software-defined-networking-hyper-v-part1.html, aufgerufen am 07.06.2014, März 2014.

[23] J. Reich, C. Monsanto, N. Foster, J. Rexford und D. Walker. Modular SDN Programming with Pyretic. *USENIX ;login*, 38(5):40–47, Oktober 2013.

[24] SearchSDN. Northbound API guide: The new network application. http://searchsdn.techtarget.com/guides/Northbound-API-guide-The-rise-of-the-network-applications, aufgerufen am 07.06.2014.

[25] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov und R. Smeliansky. Advanced Study of SDN/OpenFlow Controllers. In *Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia*, CEE-SECR '13, Seiten 1–6, New York, NY, USA, 2013. ACM.

[26] The OpenDaylight Project. Technical Overview. http://www.opendaylight.org/project/technical-overview, aufgerufen am 07.06.2014.

[27] A. Vahdat. Enter the Andromeda zone - Google Cloud Platform's latest networking stack. http://googlecloudplatform.blogspot.de/2014/04/enter-andromeda-zone-google-cloud-platforms-latest-networking-stack.html, aufgerufen am 07.06.2014, April 2014.

doi: 10.2313/NET-2014-08-1_13

Namecoin

Andreas Loibl
Betreuer: Johannes Naab
Seminar Innovative Internettechnologien und Mobilkommunikation SS2014
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: andreas.loibl@tum.de

ABSTRACT
This paper gives a brief introduction into alternative naming systems and how they can prevent and circumvent censorship. It explains the concepts of the peer-to-peer (P2P) based cryptocurrency Bitcoin and how it can be extended to implement an alternative naming system (Namecoin) without a need for a trusted authority. Namecoin domains are stored in the distributed Namecoin blockchain, a public ledger of every Namecoin transaction ever executed. It does not integrate directly in the regular domain name system (DNS), in order to resolve such domains a browser addon/extension is required. Problems like this lack of integration and usability have prevented Namecoin from being widely used currently.

Keywords
Namecoin, Bitcoin, DNS, NMC, BTC

1. INTRODUCTION
Within the current DNS structure the server operators have the power to censor information by seizing or redirecting domain names. For example, in the weeks before the local elections the Turkish government blocked Twitter and Youtube by redirecting their DNS records [1]. Another example is China, where DNS cache poisoning is commonly used to censor domains for their citizens [2].

One possible way to prevent and circumvent censorship like this is Namecoin, a distributed, secure and trustless domain name system that has no central authority.

In section 2 the paper presents the properties of naming systems and cryptocurrencies. Section 3 gives an introduction to Namecoin and explains how to register domains within the Namecoin namespace, what those domains can be used for and compare this system to regular DNS and other alternative naming systems. The paper concludes with section 4 after outlining problems and usage statistics of the current Namecoin implementation and in section 5 the referenced materials are listed.

2. ZOOKO'S TRIANGLE
Zooko Wilcox-O'Hearn [3] claims that any naming system can only fulfill two of the following three desirable properties:

1. Secure, i.e. globally unique: There is only one, unique and specific entity to which the name maps. Nobody can successfully pretend to be the owner of someone else's domain name.
2. Decentralized: There is no centralized authority for determining the meaning of a name.
3. Human-meaningful: Names are arbitrarily chosen strings short enough for humans to memorize.

This leaves three possible choices to implement a naming system [4]:

- Compromise decentralization: e.g. DNSSEC is secure and human-meaningful, but not decentralized. It is implemented with digitally signed records for DNS lookups using public-key cryptography.
- Compromise human-readability: e.g. Tor .onion and Bitcoin addresses are secure and decentralized, but not human-meaningful, because the name is just a hashed representation of a public key.
- Compromise security/ integrity: petname systems (e.g. locally defining names in `/etc/hosts`) are human-meaningful and decentralized, but not secure, as locally defined names are not globally unique.

2.1 Squaring Zooko's triangle using Bitcoin
In January of 2011 Aaron Swartz published an idea called "Squaring The Triangle" [5] in which he disproved Zooko's theory by designing a naming system that fulfills all three properties:

A naming system based on Bitcoin which uses Bitcoin's distributed blockchain as a proof-of-work to establish consensus of domain name ownership.

2.1.1 Bitcoin blockchain
The blockchain is the main invention of Bitcoin [6]. It is a transaction database that contains every transaction ever executed. Every client connected to the Bitcoin P2P network maintains a full copy of the blockchain and thus can find out how many Bitcoins belong to an address at any point in history by traversing the chain of transactions.

A transaction is a signed message that transfers certain amounts of Bitcoins from one or more previous transactions to one or more Bitcoin addresses. Each Bitcoin address is a

doi: 10.2313/NET-2014-08-1_14

representation of the public key of a private-public keypair. The private key of such a keypair is used to sign transaction messages that transfer Bitcoins from the corresponding Bitcoin address. The signed transaction is then broadcasted to every node of the Bitcoin network. Everyone can then verify the authenticity of the transaction using the public key from the sender's address.

These transactions are collected into blocks. A new block includes several recent transactions that have not yet been merged into the blockchain (so-called "unconfirmed transactions"). Once a block is created it is broadcasted and appended to the blockchain and thus becomes part of the Bitcoin history, which means that it will never be changed or removed again (so the transactions of this block are "confirmed transactions" now). Every block in the blockchain contains a reference to its previous block, thus creating a chain from the first block (genesis block) to the current one. The block-reference is a cryptographic hash of the previous block. This ensures the integrity of the chain, as any modification to a block would result in a different hash for the block and thus the reference in the next block would change, resulting in a different hash for every block after.

2.1.2 Proof of work

The creation of a block and appending it to the blockchain is called mining. This is a purposely expensive process that requires solving a unique and difficult math problem so that the number of blocks mined each day remains steady. The math problem to solve is used as a proof-of-work, as it is easy to check whether a solution is valid, but it is difficult to find a solution, as this requires a lot of trial and error. In Bitcoin the proof-of-work scheme is SHA-256, which means that the SHA-256 hash of a block's header must be lower than or equal to a specific target value in order for the block to be valid.

2.2 From Bitcoin to Namecoin

Namecoin is based on the code of Bitcoin, it uses the same proof-of-work algorithm and is limited to 21 million coins, but it has its own blockchain beginning with a different genesis block and thus it is a separate currency. In comparison to Bitcoin it has implemented additional RPC (remote procedure call) commands that allow its users to record and transfer arbitrary names (keys) and attach data (values) to those keys in the blockchain by sending special transactions. Those keys are secure and decentralized (as they are stored in the blockchain, so every node can check the validity of the operations on a key) and globally unique and human-meaningful (as they can be arbitrarily chosen), so they are fulfilling all three properties of Zooko's triangle and thus are allowing Namecoin to act as a decentralized naming system.

3. NAMECOIN

The Namecoin blockchain was started on April 18, 2011 and since then on average every 10 minutes a new block is being added. Initially the miners are rewarded with 50 Namecoins for each block, with the reward being halved every 210,000 blocks (approx. every 4 years).

In order to participate in the Namecoin currency a node needs to run a Namecoin client that has a full copy of the Namecoin blockchain and keep it in sync with the P2P network by fetching and validating new blocks from connected peers. The official reference implementation of such a Namecoin client is `namecoind` [10], which can be run in the background as daemon. It automatically connects to the Namecoin network and downloads the blockchain. It also holds the user's wallet, which contains the private keys of the keypairs of the user's Namecoin addresses. The `namecoind` daemon can be controlled by HTTP JSON-RPC commands or via the command-line, e.g.:

```
$ namecoind -daemon
namecoin server starting
$ namecoind getbalance
1.00000000
```

There are several predefined and proposed namespaces for specific use cases as listed in Table 1. For example, by registering a name in the "`id/`" namespace users could use the Namecoin naming system to create online identities and with the help of NameID [14] turn this Namecoin identity into an OpenID, which can then be used to sign into millions of OpenID-enabled websites. Namecoin's biggest and most popular namespace is however the Domain namespace "`d/`" which can be used to register and manage domain names for the virtual top level domain (TLD) "`.bit`".

3.1 Domain namespace

The following sections will describe how the Namecoin system can be used to register a domain in the Namecoin Domain namespace, and how those domains can be accessed with browsers.

3.1.1 Registering a domain

A `.bit` domain can be registered using the Namecoin RPC commands `name_new` and `name_firstupdate`, e.g. in order to register the domain `example.bit` the name `d/example` has to be pre-ordered first using `name_new`:

```
$ namecoind name_new d/example
[
  "7004db3cfda8c09945e00b5a793...",
  "abf1c2b6a64c1575"
]
```

This will reserve the domain name by sending a transaction with a hashed version of the domain name, salted with a random value (in this example the random salt is "`abf1c2`..." for transaction ID "`7004db`...").

```
$ namecoind gettransaction 7004db3cfda8c09945...
{
  "amount" : -0.01000000,
  "fee" : 0.00000000,
  "confirmations" : 0,
  "txid" : "7004db3cfda8c09945e00b5a793...",
  "time" : 1400012345,
  "details" : [
    {
```


doi: 10.2313/NET-2014-08-1_14

Namespace	Application	Status
d/<domain>	Domain names for .bit TLD	active
id/<identity>	Public online identity system (e.g. addresses for BTC, NMC, email, ...)	active
p/<personal>	Personal namespace for PGP, SSL, identities, etc.	draft
m/<message>	Messaging system for Namecoin users	draft
a/<alias>	Alias system to map a name to another address	draft
tor/<domain>	Domain names for .tor TLD for onion websites	draft

Table 1: Namecoin namespaces [9]

```
      "account" : "",
      "address" : "name_new: 13d047d3394...",
      "category" : "send",
      "amount" : -0.01000000,
      "fee" : 0.00000000
    }
  ]
}
```

The additional pre-order step is implemented in order to prevent others from stealing the domain by quickly registering the domain for themselves while the transaction for the domain registration is still unconfirmed, i.e. waiting to be included in one of the next blocks. This is why `name_firstupdate` will only be accepted after a mandatory wait period of 12 additional blocks on the blockchain after the corresponding `name_new`.

Depending on the network activity waiting for 12 blocks generally takes some time between 2 hours and 2 days. Afterwards the domain registration can be finalized using the random value and the transaction ID from the output of the previous `name_new` command and initialized with data. Arbitrary data can be stored for Namecoin keys, but in order to be able to resolve a `.bit` domain the data has to be JSON-encoded according to the schema defined in the Namecoin Domain Name Specification [8], e.g. to resolve a domain to an IP address:

```
$ namecoind name_firstupdate d/example \
> abf1c2b6a64c1575 7004db3cfda8c09945e00b5a793... \
> '{"ip":"127.0.0.1"}'
8b21511aa033ff4f5e219f548df...

$ namecoind gettransaction 8b21511aa033ff4f5e...
{
  "amount" : -0.01000000,
  "fee" : 0.00500000,
  "confirmations" : 0,
  "txid" : "8b21511aa033ff4f5e219f548df...",
  "time" : 1400023456,
  "details" : [
    {
      "account" : "",
      "address" : "name_firstupdate: d/example",
      "category" : "send",
      "amount" : 0.00000000,
      "fee" : -0.00500000
    }
  ]
}
```

In case of a name collision of the `name_firstupdate` command the one with the older `name_new` transaction wins. After the `name_firstupdate` transaction has been added to the blockchain the domain name is registered and valid:

```
$ namecoind name_list
[
  {
    "name" : "d/example",
    "value" : "{\"ip\":\"127.0.0.1\"}",
    "address" : "N1omps5uMutW9xx3gxRmwBaa...",
    "expires_in" : 36000
  }
]
```

Names and values are attached to special coins with a value of 0.01 NMC (Namecoins), stored in the user's wallet (i.e. the user owns the domain). A domain will expire 36,000 blocks (approx. 250 days) after its registration or last update. Updates are performed by creating a transaction with the name's previous coin as input. The Namecoin core code prevents those coins from being used for normal payments [7]. The domain can be updated using the `name_update` command to change its value or reset its expiration period:

```
$ namecoind name_update d/example
'{"ip":"127.0.0.1"}'
```

3.1.2 Resolving a domain

The top level domain `.bit` which is used for the domain names stored in the Namecoin blockchain is a virtual TLD that is not sanctioned by the ICANN (Internet Corporation for Assigned Names and Numbers) and thus is not assigned in the DNS root zone, so regular DNS servers are unable to resolve queries for `.bit` domains and will return a `NXDOMAIN` error instead.

There are various ways how users can reconfigure their systems and/or install additional software in order to be able to lookup `.bit` domains. A comparison of different Namecoin domain lookup software is shown in Table 2.

NMControl. NMControl is a daemon written in python that (amongst other features) acts as a local DNS server. The aim of this software is to allow developers to easily build services on top of Namecoin. When using its DNS server to resolve `.bit` domains it doesn't require trusting any third-party, all data is verified against the local Namecoin blockchain data. Therefore it requires a `namecoind` to be running in the background that has a full copy of the

doi: 10.2313/NET-2014-08-1_14

	local copy of blockchain	access .bit websites (HTTP)	DNS lookup	TLS verify (HTTPS)	supported Operating Systems and Software
FreeSpeechMe	✓	✓		✓	Firefox on Windows / Linux
NMControl	✓	✓	✓		Windows / Linux
MeowBit	✓	✓	✓		Windows
outside DNS suffix		✓	✓		any
outside DNS resolver		✓	✓		any
outside proxy		✓			any

Table 2: Comparison of some Namecoin domain lookup solutions

blockchain in sync with the Namecoin network. NMControl works on Linux and on Windows and is compatible with all Internet applications, not just a web browser. The user has to change the DNS settings of his operating system manually to use NMControl's local DNS server when it is running in order to be able to resolve `.bit` domains.

MeowBit. MeowBit is a Windows application that is based on NMControl. It adds a GUI that manages the included NMControl daemon. The primary difference is that it is easier to use for the end user, as it automatically hooks and unhooks the local name server to the DNS settings of the operating system. Another advantage for the end user is that it is available as a single installation bundle ("`setup.exe`"), so the Windows user doesn't have to take care of installing Python, fetching the NMControl source code, configuring its DNS plugin, etc. It only requires to have the Namecoin wallet software running in the background, which can simply be downloaded as a pre-compiled Windows binary from the official Namecoin website.

FreeSpeechMe. FreeSpeechMe is a browser extension that only works with the Mozilla Firefox browser for Linux and Windows operating systems. It can resolve `.bit` domains to IP addresses from the IP address mappings stored in the Namecoin blockchain. It retrieves the blockchain data via NMControl, therefore it requires NMControl and a Namecoin wallet software (or `namecoind`) to be installed and running in the background. Unlike plain NMControl the FreeSpeechMe browser extension doesn't use NMControl's DNS server to resolve `.bit` domains, so the user doesn't have to touch the DNS settings of his operating system to be able to browse `.bit` domains. Otherwise it implements the same security properties as NMControl, i.e. it does not require trusting any third-party. Additionally it verifies `.bit` TLS certificates against fingerprints stored in the Namecoin blockchain, which allows safe usage of self-signed certificates for browsing `.bit` websites via HTTPS without trusting any third party. This makes FreeSpeechMe one of the most secure ways to browse the `.bit` namespace [11].

Lookup for outsiders. To be able to access `.bit` websites when using systems that currently are unable to run the Namecoin software locally (e.g. mobile devices) some fallback solutions are possible. However using these workarounds permanently or for anything sensitive is strongly discouraged as it introduces a middle man that could possibly censor, hijack or surveil the internet traffic and has to be trusted completely.

One possible solution would be to point the domain search suffix to a publicly available Namecoin DNS suffix gateway. For example if the Namecoin DNS suffix gateway is available through the domain "`namecoin-suffix.dot-bit.org`" it could provide DNS resolution for `.bit` domains with its domain name as suffix: `example.bit.namecoin-suffix.dot-bit.org` would be resolved by the "`namecoin-suffix.dot-bit.org`" server to the IP address of `example.bit` that is stored in the Namecoin blockchain.

The user would then simply add "`namecoin-suffix.dot-bit.org`" to the DNS search domains in his network settings. If he tries to visit `http://example.bit/` his system will query the configured DNS server for `example.bit`, but the DNS server will return `NXDOMAIN`, as the `.bit` TLD doesn't exist in the root zone. Then his system will append the search domain and query the server again for `example.bit.namecoin-suffix.dot-bit.org` which in turn will query the DNS server of `namecoin-suffix.dot-bit.org` who is able to resolve the domain (as explained above). The downside of this approach is that it is insecure: the operator of the Namecoin DNS suffix gateway receives all NXDOMAIN queries, so he is able to log and inject false answers or hijack NXDOMAINs.

Another possibility would be to use a (public) DNS server that resolves `.bit` domains as resolver. There are some volunteer-run DNS servers that can be used for testing purposes, e.g. `dns.dot-bit.org` With this solution the operator of the DNS server will receive all DNS queries, which means that the operator can monitor all queries and fake answers to redirect or hijack any domain, so it is even worse than the DNS suffix solution regarding security.

Finally it is also possible to use a full proxy service that allows HTTP(S) access to `.bit` websites. This does not require any additional software or network setting changes on the client. Regarding security this is the worst solution, as the operator of the proxy server not only can monitor, fake and hijack domains, he also can receive and modify all traffic of the `.bit` website. There are some publicly accessible `.bit` web-proxies, e.g. [12, 13].

3.1.3 Comparison to regular DNS and other alternative name systems

In contrast to the decentralized Namecoin naming system the regular DNS system is hierarchical and centralized, with the Internet Assigned Numbers Authority (IANA) as overall authority for the IP addresses, the domain names, and many other parameters, used in the internet [15]. The ICANN acts as IANA and is responsible for management of the DNS root

doi: 10.2313/NET-2014-08-1_14

zone (“.”), the top-level zone of all domain names. It assigns the operators of the top level domain and ensures the maintenance and the administrative details of the individual TLDs. Verisign serves as the root zone maintainer under a cooperative agreement entered with the United States government [15, 17].

Namecoin doesn't have or need such a hierarchy of authorities, all domain records are publicly known to everyone as they are stored in the publicly shared and distributed Namecoin blockchain.

Regular DNS and Namecoin also have fundamentally different business models, DNS registrars have to make a profit from registrations whereas the OpenSource Namecoin project is community based and not profit oriented. This is reflected in a substantial difference in pricing:

Prices for a regular DNS domain name registration from an ICANN accredited registrar (e.g. for the Verisign “`.com`” TLD) are around $ 10.00 per year. The costs for registering domains in Namecoin vary depending on the NMC/USD exchange rate and on the transaction fees for the Namecoin network:

For every name operation (`name_new`, `name_firstupdate`, `name_update`) the Namecoin transaction fees have to be paid to the miners, otherwise the transaction is unlikely to be included into the blockchain. Currently the standard transaction fees for Namecoin are 0.005 NMC/transaction. The command `name_new` to pre-order a domain has an additional fixed cost of 0.01 NMC. The command `name_firstupdate` initially (at the genesis block) had a network fee of 50 NMC, slowly decreasing over time. This was implemented in order to prevent domain squatting in the early days of Namecoin. Since December 2012 the network fee has fully decreased, so `name_firstupdate` is free today (except for the transaction fees). So the costs to register a new `.bit` domain for one year are 0.015 NMC (`name_new`) + 0.005 NMC (`name_firstupdate`) + 0.005 NMC (`name_update` to prevent it from expiring after about 250 days) = 0.02 NMC. At the current exchange rate of $ 1.79 per NMC (see Table 3) this results in just a few cents per year (approx. $ 0.036). Of course this is just a rough estimate based on the rates at the time of writing - the exchange rates are always fluctuating and might rise or fall by several orders of magnitude when Namecoins are traded on the markets.

Tor hidden services. Tor is an open-source anonymous communication service that implements a practical design for “hidden services”, which allows its users to offer services like a webserver without revealing their IP address. In order to connect to such a Tor hidden service an alternative naming system with the pseudo-top-level domain `.onion` is used: Addresses in the `.onion` pseudo-TLD are not human-meaningful, they are an alpha-semi-numeric, hashed representation of the public key used by the hidden service, e.g.: `http://3g2upl4pq6kufc4m.onion/`. Such addresses are not actual DNS names, and the `.onion` TLD is not included in the DNS root zone, only with the Tor proxy software installed Internet programs like browsers can access sites with `.onion` addresses by sending the request through the network of Tor servers.

3.2 Problems

One of the problems of Namecoin is the so-called “domain squatting”: Because the cost of registering domains is so cheap those who own Namecoins can register thousands of `.bit` domains anonymously and hold on to forever, as there is no authority instance that could handle disputes and enforce a domain transfer to take it away from them. Many people are using this opportunity to buy up trademarked or copyrighted names [16].

Currently mobile support is also an issue with Namecoin domain name system - there is no support for mobile devices like Android- or iOS-smartphones, as those would somehow have to run a full Namecoin node in the background and keep a local copy of the whole, ever growing blockchain in sync with the Namecoin network, just to be able to lookup `.bit` domains. This requires a substantial amount of resources like memory, storage space and (mobile) network traffic that is not suitable for the hardware of current mobile devices. Additionally the locked-down APIs of the mobile operating systems make it difficult to make Namecoin domain lookups usable for all applications, e.g. an Android phone would have to be rooted to be able to change the system's DNS resolver address to a different, `.bit`-capable (local) DNS server.

Another possible source of problems is the code quality of the Namecoin core, e.g. in October 2013 a severe bug was found in the name verification code that allowed everyone to take an already registered name and update it with a new value, even though this name did not belong to them. There was no enforced integrity of the key value pairs in Namecoin before version Q.3.72, which basically defeated the entire purpose of Namecoin. This was quickly fixed and a new Namecoin version was released the next day, resulting in a hardfork of the Namecoin blockchain after block 150,000 [18]. This means that there was an out of bound agreement to change the Namecoin name verification rules, so that transactions overwriting the names of others won't be accepted any more after this block.

Namecoin shares all structural theoretical weaknesses with Bitcoin, e.g. if an Attacker has a lot of computing power and controls more than 50% of the whole Namecoin network's mining power it allows him to exclude and modify the ordering of transactions. Thus he will be able to block transactions from being confirmed and prevent other miners from mining any valid blocks while he is in control. The attacker can also change the (recent) blockchain history: he can modify an existing block and start mining a blockchain fork that builds on top of the modified transaction. Since the attacker has more computing power than the rest of the network combined he can generate blocks faster than the rest of the network. Eventually his private fork will become longer than the real blockchain branch. When this happens his fork will become the new “official” blockchain per definition, as it is the longest branch now. The rest of the network will switch to his branch and continue mining there. This attack could be used for double-spending coins or (Namecoin specific) to steal a domain registration:

1. Modify the block that contains the `name_firstupdate` transaction of the domain to steal:

doi: 10.2313/NET-2014-08-1_14

(a) Remove the `name_firstupdate` transaction.

(b) Add own `name_new` transaction that pre-registers the domain for the attacker.

⇒ This modification creates a private fork of the blockchain.

2. Mine at least 12 new blocks on top of the private fork without including the previously removed `name_firstupdate` transaction again.

3. Add own `name_firstupdate` transaction with the matching random value of the pre-registration from step 1.(b) into the next block.

4. Continue mining blocks for the private fork until it becomes the longest branch in the network.

5. Now the domain is registered to the attacker.

6. The real, previously removed `name_firstupdate` transaction is now invalid, as the attacker's registration took its place.

3.3 Statistics

Following are some facts and statistics about Namecoin:

- According to Table 3 Namecoin is currently the 6. biggest mineable cryptocoin.
- The exchange rate for NMC/USD fluctuated by a factor of 8 during a period of less than 6 months
- Up until Aug 3, 2014 more than 9.4 million Namecoins have been created [20].
- The Namecoin blockchain contains more than 190,000 blocks.
- The full blockchain and its index files require more than 1,900 MB to be stored.
- Currently there are 122,772 non-expired names registered in the "`d/`" namespace (TLDs `.com`: 113,639,892 / `.de`: 15.743.799)
- 108,114 (88%) of those are valid `.bit` domain names.
- 6,790 (5.5%) of those can be resolved to a website [20].
- 429 (0.3%) of those are unique (mapped to unique IP addresses) [20].

4. CONCLUSION

This paper has outlined how the Bitcoin-based Namecoin system can be used as an alternative to the hierarchical and centralized DNS and how this could potentially prevent and circumvent censorship. However the practical use cases are currently limited by several problems, which makes this system unlikely to be adopted by a signifcant number of internet users at this time. Due to this nicheness even more problems arise, like the large amount of name squatting, as only 0.3% of all currently registered domain names can be resolved to unique websites.

#	Name	Market Cap	Price per coin
1	Bitcoin	$ 7,305,063,919	$ 566.380000
2	Litecoin	$ 286,957,784	$ 9.800000
3	Darkcoin	$ 44,446,712	$ 10.140000
4	Peercoin	$ 36,504,715	$ 1.700000
5	Dogecoin	$ 31,191,323	$ 0.000378
6	**Namecoin**	**$ 16,161,798**	**$ 1.790000**
7	BitShares-PTS	$ 6,951,251	$ 4.230000
8	NeutrinoCoin	$ 6,914,311	$ 0.032882
9	Quark	$ 4,056,502	$ 0.016361
10	Vertcoin	$ 3,767,091	$ 0.719721

Table 3: Top10 Crypto-Currency Market Capitalizations [19] (Jun 15, 2014 7:15 AM UTC)

5. REFERENCES

[1] *Turkey's YouTube and Twitter bans show a government in serious trouble.* Alev Scott, The Guardian, March 2014. `http://www.theguardian.com/commentisfree/2014/mar/28/turkey-youtube-twitter-ban-government-trouble`

[2] *DNS Cache Poisoning in the People's Republic of China.* ViewDNS.info, September 2011. `http://viewdns.info/research/dns-cache-poisoning-in-the-peoples-republic-of-china/`

[3] *Names: Distributed, Secure, Human-Readable: Choose Two.* Zooko Wilcox-O'Hearn, 2001. `https://web.archive.org/web/20011020191610/http://zooko.com/distnames.html`

[4] *Zooko's triangle.* Wikipedia, 15.06.2014. `http://en.wikipedia.org/wiki/Zooko's_triangle`

[5] A. Swartz. *Squaring the Triangle: Secure, Decentralized, Human-Readable Names*, Jan 2011. `http://www.aaronsw.com/weblog/squarezooko`

[6] S. Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System.* `https://bitcoin.org/bitcoin.pdf`

[7] *Namecoin FAQ.* `https://dot-bit.org/FAQ`

[8] *Namecoin Domain Name Specification*, version 2.0. `https://wiki.namecoin.info/?title=Domain_Name_Specifica`

[9] *Namecoin Namespaces.* `https://github.com/namecoin/wiki/blob/master/Namespace.`

[10] *namecoind*, sourcecode of the Namecoin-client reference implementation. `https://github.com/namecoin/namecoin`

[11] *FreeSpeechMe.* Software plug-in for Firefox that allows people to effortlessly view `.bit` websites. `https://directory.fsf.org/wiki/FreeSpeechMe`

[12] *dotbit.me proxy.* `https://dotbit.me/proxy/`

[13] *bit.no.com.* namecoin web proxy. `http://bit.no.com/`.

[14] *NameID.* Crypto-OpenID. `https://nameid.org/`.

[15] J. Postel. Rfc 1591, Domain Name System Structure and Delegation, 1994.

[16] J. Prasanna, S. Prakash, Nikhil, S. Selvan. *.bit squatting*, Cyber Security & Privacy Foundation, Jan 2014. `http://cybersecurityprivacyfoundation.org/Namecoin_bit.pdf`

[17] *Root Zone*, ICANN Wiki, 15.06.2014. `http://icannwiki.com/index.php/Root_Zone`

[18] *Namecoin was stillborn, I had to switch off life-support*, Michael Grønager, Payward Inc., 15.10.2013.

https://bitcointalk.org/index.php?topic=310954.0
[19] *Crypto-Currency Market Capitalizations.*
http://coinmarketcap.com/
[20] dotbit.me: *The simple way to a .bit address.*
https://dotbit.me/
[21] BitcoinWisdom BTC-e NMC/USD.
https://bitcoinwisdom.com/markets/btce/nmcusd

doi: 10.2313/NET-2014-08-1_14

A Look at Intel's Dataplane Development Kit

Dominik Scholz
Supervisors: Daniel Raumer, Florian Wohlfart
Seminar Innovative Internettechnologien und Mobilkommunikation SS 2014
Chair for Network Architectures and Services
Department of Informatics, Technische Universität München
Email: scholzd@in.tum.de

ABSTRACT

The increased performance and cost-efficiency of modern multi-core architectures allows for packet processing implemented in software instead of using dedicated hardware. Such solutions provide large flexibility as features can be added at any time. The drawback of such systems are bottlenecks which appear during the processing of packets at line rate up to 10 Gbit/s. Hence, specialized high-speed packet processing frameworks are needed to use the full potential of commodity hardware. Different research groups have identified several limiting factors and proposed their own solutions by implementing custom drivers and kernel modules. We provide a survey-like introduction to three different frameworks: the Intel Dataplane Development Kit, netmap and PF_RING DNA. To understand the required background of packet processing implemented in software, we explain the mechanisms of modern network adapters and network stacks by reference to Unix-based operating systems. In particular, we present and compare the different techniques that are used by those frameworks to make high-speed packet processing feasible.

Keywords

commodity hardware, highspeed packet processing, Linux network stack, Intel Dataplane Development Kit, netmap, PF_RING DNA

1. INTRODUCTION

Through steady development the performance of commodity hardware has been continuously increased in recent years. Multi-core architectures have therefore become more interesting to fulfil network tasks by using software packet processing systems instead of dedicated hardware. As an example, all Unix-based operating systems can accomplish network tasks, as a full protocol stack that handles the packet processing, is implemented. The advantages of high flexibility and lowered costs are contrasted with possible higher performance and reduced energy consumption achieved with specialized hardware [21, 16].

Unix-based Software routers are a good example of packet processing done using commodity hardware. Features can be rapidly deployed, whereas dedicated hardware would consume extensive development cycles before being able to be upgraded. The RouteBricks project has shown that software routers are in general capable of reaching the same performance that hardware routers deliver today [22].

In order to achieve the line rate of 1 Gbit/s and 10 Gbit/s network adapters, the problems of common packet processing software have to be identified. Various research groups [21, 22, 23] have therefore created models to determine those limitations by running tests for different use cases. They come to the conclusion that standard software is not capable of reaching the maximum performance for small packet sizes. Hence, several projects developed frameworks for high-speed packet processing using commodity hardware [24, 25, 26]. These frameworks replace or extend conventional concepts by implementing their own driver and kernel level improvements. Some of the concepts used by these groups are explained in this paper.

Other research projects created software solutions that specialize on specific tasks like bridging or routing. One example is Open vSwitch [27], an OpenFlow switch that can fully replace a hardware switch. An instance of the latter is Click modular router [28].

The remainder of this work is structured as follows: in chapter 2 we provide the background for packet processing using commodity hardware. In particular, we use the Linux network stack as an example, to highlight the arising problems with high-speed packet processing in common software implementations. Chapter 3 presents the Intel Dataplane Development Kit (DPDK) and explains the main concepts. Then, chapter 4 compares the Intel DPDK with two other packet processing frameworks, netmap and PF_RING DNA. Chapter 5 concludes this paper.

2. PACKET PROCESSING USING COMMODITY HARDWARE

In this chapter we give an overview of a common software implementation for packet processing using commodity hardware. For this purpose we use the network stack of Unix-based operating systems as an example. We will begin with the tasks of the network interface card (NIC) that receives incoming packets, followed by the Linux network stack that implements the necessary dataplane functionalities and conclude with the transmission of the network packet via a NIC. An abstract view of this process is illustrated in figure 1. The resulting performance limiting factors that can be identified during the packet processing of this implementation conclude this chapter.

2.1 Receiving Side NIC

With the arrival of the packet at the receiving NIC (1) two tasks have to be accomplished [2, 3]. Firstly, the packet has to be transferred to main memory (2). Therefore the Linux kernel uses the *sk_buff* structure for the internal representation of packets. The NIC has a buffer, usually a ring

doi: 10.2313/NET-2014-08-1_15

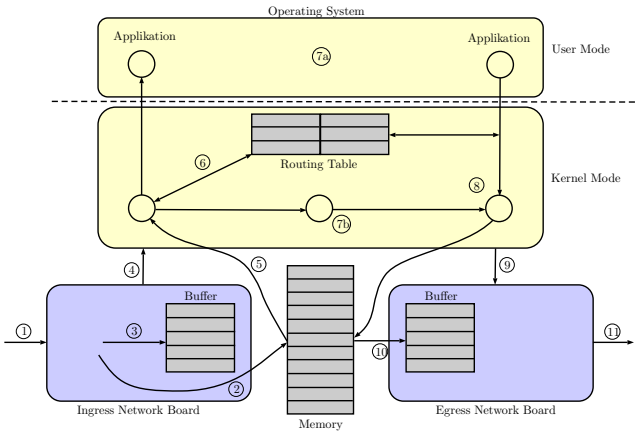


Figure 1: Abstract model of the packet processing steps in Unix-based software routers

queue called 'ring buffer', to store pointers to *sk_buff* structures (3), one for each packet reception and transmission [3]. When receiving a packet, one of these descriptors will be used, assuming that it is ready. This means that the descriptor has to be initialized and allocated with an empty *sk_buff*. To achieve this the direct memory access (DMA) engine of the NIC is used, mapping the structure to kernel memory space. If this was successful the packet is stored in a *sk_buff* and is ready for further processing, otherwise it will be discarded by the NIC, because no packet descriptor was available [4].
The second step is to inform the kernel that a packet is available (4). The NIC schedules a hardware interrupt and eventually the CPU will respond by calling the interrupt handler of the driver [3]. Since the kernel version 2.4.20 the driver uses the New API (NAPI) [2]. The interrupt handler adds the NIC to a so called *poll_list* and schedules a soft interrupt. When this interrupt gets served the CPU polls each of the devices present in the list to get their available packets from the ring buffer. Then, for each of these packets a function is called to start the processing through the network stack [3].

2.2 Linux Network Stack

The network stack processes each packet layer by layer [1, 3, 4], starting with the network layer. At first, basic checks, including integrity-verification and application of firewall rules, are performed. If the packet is faulty it gets dropped, otherwise it gets passed to the routing subsystem (6) in order to make a decision whether the packet has to be delivered locally to a userspace application or forwarded to another host. This decision is being made based on the implemented routing algorithm that finds the best match for the destination IP address of the packet in the routing table [1].
If this address equals one of the hosts locally configured addresses the packet has to be delivered locally and is handed over to the transport layer. The Linux network stack provides a complete implementation of the TCP and UDP protocols. A more detailed description of this implementation is given in [3]. After passing the transport layer the packet can finally be delivered to the application. Using the socket API [19] the data of the *sk_buff* gets copied to userspace. Now the application has full access to the data of the received packet (7a). The same way an application in userspace can pass a packet down to the network stack in order to transmit it (8). Here, one of the main tasks is to determine the next hop and corresponding interface for an outgoing packet. This is accomplished by looking up the destination IP address in the routing table [3]. In the case of forwarding the packet (7b), no processing of layers beyond layer 3 is needed. On the network layer the main task is to decrement the TTL header field and, in case it reached zero, to drop the packet and send a corresponding ICMP message [1]. Furthermore, sanity checks, firewall rules and fragmentation can be applied. Afterwards the packet gets passed to the transmitting part of the network stack, too (8).
The layer 2 processing does not differ for forwarded and locally created packets. Based on the result of the routing lookup, the layer 2 address of the next hop has to be identified using for instance the Address Resolution Protocol. After meeting these tasks a function of the NIC is called, informing it to transmit the packet.

2.3 Transmitting NIC

To finally transmit a packet the transmitting NIC has to fulfil two tasks [3]. First of all the driver has to load the packet descriptor, which holds the location of the *sk_buff* in main memory, into the transmitting ring buffer (10). Afterwards he tells the NIC that there are packets available and ready to send (11). Secondly the NIC has to inform the CPU via an interrupt that the *sk_buff* structure can be deallocated.

2.4 Performance Limiting Factors

The Linux network stack is designed for general purpose networking. It is not only able to function as a router, but also supports various protocols on different layers like IPv4 and IPv6 on the network layer or TCP and UDP on the transport layer [3]. While this design choice is convenient for running common applications for "normal" users up to a rate of 1 Gbit/s, it rapidly reaches a limit when approaching the 10 Gbit/s at which the operating system can't handle more packets and thus starts dropping them. The following paragraphs list and explain some of the impediments which are being addressed by the frameworks presented in sections 3 and 4.3 [16].
The identified bottleneck is the CPU. A CPU can only operate a limited number of cycles per second. The more complex the processing of a packet is, the more CPU-cycles are consumed, which then can not be used for other packets. This limits the number of packets per second. Therefore, in order to achieve a higher throughput, the goal must be to reduce the per packet CPU-cycles.
Another limiting factor is the usage of the main memory, which can be further divided into three problems: per-packet allocation and deallocation, complex *sk_buff* data structure and multiple memory copies. The first one occurs right after the packet has been received by the NIC as described in section 2.1. For every packet a *sk_buff* has to be allocated and later on, when the packet has been passed to user-level or has been transmitted, deallocated again. This behaviour leads to excessive consumption of bus-cycles - that is, CPU-cycles spent transferring data from the main memory to the CPU - by the CPU, causing significant overhead. The second problem arises from the effort of the network stack to be compatible with as many protocols as possible. Therefore

doi: 10.2313/NET-2014-08-1_15

```
struct sk_buff {
  [...]

  /* Transport layer header */
  union
  {
          struct tcphdr *th;
          struct udphdr *uh;
          struct icmphdr *icmph;
          struct igmphdr *igmph;
          struct iphdr *ipiph;
          struct spxhdr *spxh;
          unsigned char *raw;
  } h;

  /* Network layer header */
  union
  {
          struct iphdr *iph;
          struct ipv6hdr *ipv6h;
          struct arphdr *arph;
          struct ipxhdr *ipxh;
          unsigned char *raw;
  } nh;

  [...]
}
```

Figure 2: Excerpt of the source-code of the *sk_buff* structure (image from [3])

the *sk_buff* structure (see figure 2) contains the metadata of several protocols, which may not be needed for the processing of the packet or the application. This complexity leads to an unnecessary large data structure, slowing down the processing. The latter problem arises from the fact that a packet has to traverse different stations until it reaches the application, resulting in at least two copy operations of the complete packet: after packet reception the data is copied from the DMA-able memory region to the *sk_buff* and later on it is copied from this buffer to the user-level application. In [16] is shown that 63% of the CPU usage during the processing of a 64 Byte large packet is *sk_buff*-related.

A third major problem is the context switching from user mode to kernel mode and vice versa. Every time the userlevel-application wants to receive or send a packet it has to make a system call, causing a context switch to kernel level (and later on back to user level). While this design improves the resistance to system crashes as the application in user space cannot use critical hardware functions, it consumes a huge part of the CPU-cycles [16].

When using multiple CPUs the packet processing can be slowed down because of so called spinlocks [17]. Those locks are the Linux implementation of active waiting-mutexes and appear on multiple occasions in the network stack. For example, in order to transmit a packet two locks have to be acquired, which protect the access to the NIC's transmitting queue [1], resulting in a bottleneck as possible parallel processing of frames is negated.

All of the mentioned design choices make the Linux network stack a good general purpose networking solution, but not for applications that require high-speed packet processing. Therefore, packet processing frameworks like the ones presented in the remaining chapters of this paper implement techniques that try to resolve most of the mentioned problems.

3. INTEL'S DATAPLANE DEVELOPMENT KIT

Chapter 2 showed that there is indeed need for frameworks that replace or extend the network stack to solve problems like memory allocation per packet or context switches in order to achieve the maximum performance. This chapter will present the Dataplane Development Kit from Intel (DPDK), first released in 2012. First an overview of the framework is given, afterwards the main implemented libraries are explained. This includes the libraries for queue management, memory management and packet buffers. This chapter concludes with an illustration of how to use the DPDK with user-land applications and what the inherited problems are.

3.1 Overview

The Intel Dataplane Development Kit is a framework that provides a set of software libraries and drivers for fast packet processing in dataplane applications on Intel architectures [5, 6, 8]. It supports Intel processors in 32-bit or 64-bit mode from Intel Atom to Intel Xeon generation, with or without non-uniform memory access (NUMA), with no limits to the number of cores or processors. The DPDK runs on Linux operating systems, but completely replaces the network stack, implementing a run-to-completion model: "all resources must be allocated prior to calling data plane applications, running as execution units on logical processing cores" [8]. Furthermore, scheduling is not supported as all devices are accessed via polling. This reduces the overhead produced through interrupt processing in high-speed scenarios [6]. Additionally a pipeline model may be used, which exchanges messages between different cores to perform work in stages.

As shown in figure 3, the DPDK provides the libraries, which are executed completely in userspace, by creating the Environment Abstraction Layer (EAL). The EAL hides the specifics of the available soft- and hardware and provides a interface of library functions that can now be used by ones application.

3.2 Libraries

The Intel DPDK provides several libraries that are optimized for high performance. They fulfil basic tasks, similar to what the Linux network stack does: allocating memory for network packets, buffering packet descriptors in ring-like structures and passing the packets from the NIC to the application (and vice versa). The libraries that are necessary for these assignments include memory, queue and buffer management, which are presented in the following sections. An excerpt of additional features supplied by the Intel DPDK is shown in section 3.2.4.

3.2.1 Queue Management

To manage any sort of queues the *librte_ring* library provides a ring structure called *rte_ring*, shown in figure 4. This struc-

doi: 10.2313/NET-2014-08-1_15

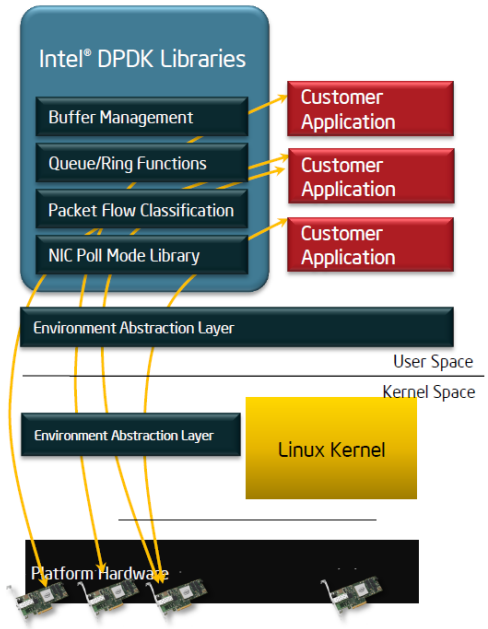


Figure 3: Overview of the Intel DPDK libraries and Environment Abstraction Layer (image from [6])

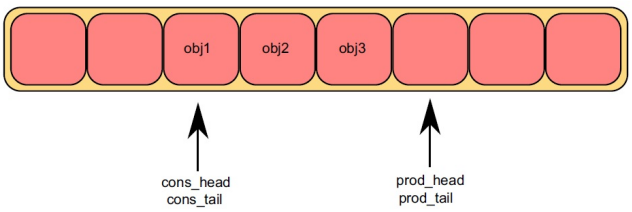


Figure 4: The *rte_ring* structure (image from [8])

ture has the following properties [8]: the ring uses a fixed-size, lockless implementation following the FIFO principle, supporting single and multi producer/consumer en- and de-queue scenarios. Furthermore, it is able to en-/dequeue a specified amount (bulk) or all (burst) available packets. The ring is implemented as a table of pointers to the stored objects and uses two head and tail couples of pointers, one for each the consumers and producers, to control the access. To fully understand the functionality of this structure and why it has the named properties one has to study several scenarios of producer/consumer interaction, therefore the details of a lockless ring-buffer are not shown in this paper (see [20] and [8] for further understanding).
Compared to a normal queue, implemented using a double-linked list of infinite size, the *rte_ring* has two big advantages. First of all it is much faster to write to this structure as the access is protected using lockless mechanisms [20]. Secondly, due to saving the pointer to data in a table, bulk and burst operations cause way less cache misses. Of course the disadvantage is the fixed size, which cannot be increased during runtime and also costs more memory than a linked-list as the ring always contains the maximum number of pointers.
The *rte_ring* can be used for communication, for example between applications, but also to manage the contents of memory pools (section 3.2.2) and to store buffers of network packets (section 3.2.3).

3.2.2 *Memory Management*

The EAL provides a mapping of physical memory [8]. It creates a table of descriptors which are called *rte_memseg* which each point to a contiguous portion of memory to circumvent the problem that available physical memory can have gaps. Those segments can then be divided into memory zones (figure 5). These zones are the basic memory unit

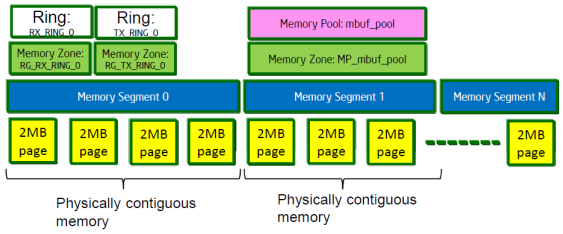


Figure 5: Distribution of physical memory into segments and zones for further use (image from [6])

used by any further object created from an application or other libraries.
One of these objects is the *rte_mempool* provided by the *librte_mempool* library. This structure (see figure 6) is a pool of fixed-sized objects that uses a *rte_ring* to store free objects and can be identified by a unique name [8].
To increase the performance, a specific padding can be

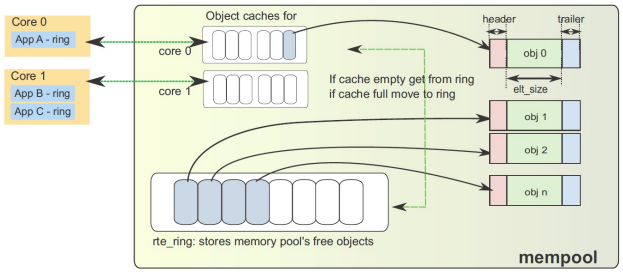


Figure 6: A *rte_mempool* with its associated *rte_ring* (image from [8])

added between objects. This ensures "that the beginning of each object starts on a different channel and rank in memory so that all channels are equally loaded" [8]. Examples, when to use this optimization, include layer 3 forwarding and flow classification of network packets, as only the first 64 bytes are necessary to make a decision [8].
Although a lockless ring is used to control the access to free objects, the costs may be high if multiple cores access the ring causing a compare-and-set operation each time. To avoid having the ring as a bottleneck, each core has an optional local cache in the *rte_mempool*. The core has full access to his cache of free objects, using a locking mechanism. When the cache reaches full capacity or is empty it uses a bulk operation to exchange free objects with the ring of the *rte_mempool*. This way the core has fast access to the

objects he potentially uses repeatedly and relieves the ring structure, which then can do operations with other cores [8].

3.2.3 Network Packet Buffer Management

For any application to be able to transport network packets, the *librte_mbuf* library provides buffers called *rte_mbuf*. These buffers are created before the actual runtime of the application and are stored in a *mempool*. During runtime the user can now use a *rte_mbuf* by specifying a *mempool* from which it wants to take an available one. Freeing a *rte_mbuf* simply means returning it back to the *mempool* it originated from [8].

The size of a *rte_mbuf* is kept as small as possible to fit

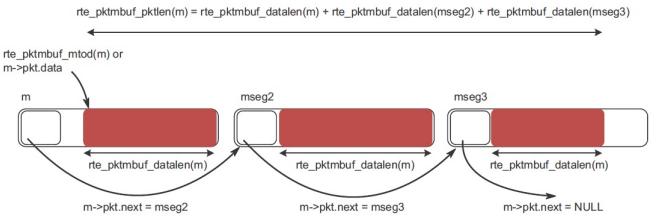


Figure 7: One packet as chain of *rte_mbuf*s. Every *rte_mbuf* contains the same metadata, a pointer to the next buffer and a part of the packet data (image from [8])

in one cache line. In order to hold the information of a larger packet, multiple *rte_mbuf*s can be chained together. As illustrated in figure 7, one *rte_mbuf* does always contain metadata, which is "information [...] retrieved by the network driver [...] to make processing easier" [8], and a part of the data of the packet itself, including header information. It also contains a pointer to another *rte_mbuf* to allow buffer chaining.

This structure can easily be extended to contain other types of information, like serving as a control buffer or for logging reasons.

3.2.4 Further Libraries and Features

Aside from the already illustrated libraries which are needed to allow basic networking operations, the Intel DPDK also has further libraries and drivers that provide specialized functionalities.

The LPM library implements a table that allows longest prefix matching (LPM), which is an algorithm used to forward packets depending on their IPv4 address [1]. The main functions are to add or delete a rule to or from the table and lookup an IP address using the LPM algorithm. The best fitting entry is then returned. Based on this library an application can be written to fulfil IPv4 forwarding tasks. The same functionalities but for the IPv6 protocol are provided by the LPM6 library [8].

Another library offers similar functionalities by providing hash functions. The hash library allows to do efficient and fast lookups in a big set of entries using a unique key[8]. This can then be used for flow classification, that is, mapping a packet to the connection, called flow, it belongs to. This is useful to match for instance TCP packets to their connection and process all the frames together in the context of this flow. A common procedure to achieve this matching is to create a hash value of the 5-tuple of the IP- and transport-layer headers (source and destination address, protocol, source and destination port) [8].

The timer library provides the ability for functions to be executed asynchronously, which can be used for example for garbage collectors. The timer can be executed with high precision only once or periodically. It uses an interface of the EAL or the CPU's Time Stamp Counter to receive a precise time reference [8].

The DPDK also includes poll mode drivers for 1 Gbit/s and 10 Gbit/s network cards and several other emulated frameworks like virtio or VMXNET3 adapter [8]. These provide APIs to directly configure the device and access the receiving and transmitting queues without the need for interrupts, comparable to the NAPI drivers.

Several more specialised libraries are also provided. See [8] for a full list and detailed descriptions.

3.3 Using the Dataplane Development Kit

The Intel DPDK provides all its libraries by creating the Environment Abstraction Layer (EAL) [5, 7]. The EAL has access to low-level resources including hardware and memory space and provides a generic programming interface that is optimized for the hardware of the system, but hides the environment specifics from the applications and libraries. The supplied services include the loading and launching of applications, reserving memory space in form of memory segments as described in section 3.2.2 and trace and debugging functions for instance for the logging of events [8].

Before being able to compile an application the Intel DPDK target environment directory, which contains all libraries and header files, has to be created. To compile an application two variables must be exported: *RTE_SDK* pointing to the installation directory of the DPDK and *RTE_TARGET* pointing to the environment directory. The compiled program can now be run by linking it with the EAL [7].

One has to keep in mind that the Intel DPDK only provides a framework for basic operations that allow packet processing at high speed. It does not provide any functionalities like layer three IP forwarding, the use of firewalls or any protocols like TCP or UDP that are fully supported by the Linux network stack. It is the programmers responsibility to build all the needed functionalities using the provided libraries. Thus, porting an application to the Intel DPDK can be time consuming.

However, the development of applications is eased as many example programs, illustrating the use of different libraries or features, are provided. Furthermore, the *dpdk.org*-project tracks the Intel DPDK by not only listing recent patches, but also contributing major enhancements provided through the community [5]. The DPDK vSwitch [9], an improved version of OpenvSwitch, is an example of an application that is based on the Intel DPDK and achieves significant performance boosts [10].

4. OTHER FRAMEWORKS

As discussed in chapter 3 the Intel DPDK is a framework to speed up packet processing by completely replacing the network stack on Unix-based operating systems and implementing different techniques like buffer pre-allocation or a custom packet buffer. Other frameworks with similar purposes have been developed over the last few years, sharing some of the approaches used in the Intel DPDK, but also

doi: 10.2313/NET-2014-08-1_15

using different ideas. This chapter will present two selected projects by shortly explaining the main concepts and techniques, without getting into the details of their respective implementations. Firstly, the netmap framework for fast packet I/O will be introduced, followed by an engine called PF_RING DNA. This chapter will conclude with a comparison of the three presented packet processing frameworks.

4.1 netmap

Netmap is a framework developed at the University of Pisa by Luigi Rizzo within "The netmap project" [11]. It is designed for fast packet I/O in a safe manner, implemented as a kernel module, thus, running completely in kernel mode. Netmap implements techniques to resolve three problems of packet processing: the dynamic allocation of memory per packet, overhead produced through system calls and multiple memory copies [12].
The data structures of the netmap architecture reside in the same memory region (illustrated in figure 8) which is shared by all user processes. This makes the forwarding of packets

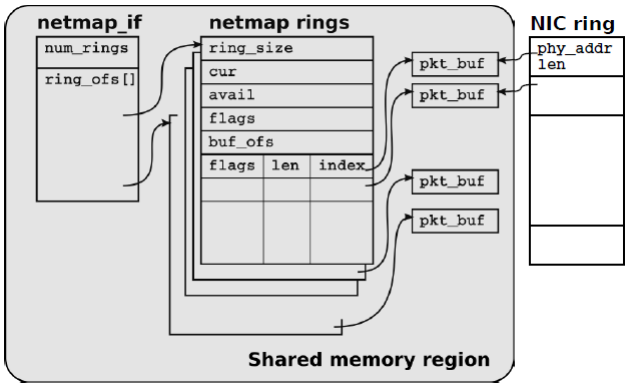


Figure 8: The structures of the netmap architecture (image from [12])

between different interfaces possible without a single copy operation. Furthermore, netmap uses its own lightweight representation of packet buffers [12]. Those *pkt_buf*-structures are fixed-sized and are preallocated before the actual packet processing begins. Each packet buffer is referenced by the ring of the corresponding NIC. Additionally, *netmap rings* are used to keep references to the buffers. This is a circular queue that uses an array to store the indices and other information of the packets. The last data structure, called *netmap_if*, contains important information, like the number of rings, about an interface. To reduce per-packet processing costs, netmap provides batching techniques. Also, a method called parallel direct paths is utilized to solve the problem of serialization during the access of traffic [16]. Basically, a direct path from a receiving or transmitting queue of the NIC to the application in user space is constructed and being assigned to only one core. This supports Receive Side Scaling (RSS) [18], a technique to distribute incoming network traffic to multiple CPUs. RSS is supported by many network cards as it "relieve[s] bottlenecks in receive interrupt processing caused by overloading a single CPU" [18].
Despite using shared memory regions and keeping the structures in kernel context, a misbehaving user application cannot cause the kernel to crash [12]. However, processes can corrupt each others netmap rings and packet buffers. A possible solution for this problem is to distribute each ring to a separate memory region.
The netmap framework supports a wide range of soft- and hardware. Initially it was developed for FreeBSD operating systems, but a Linux-version has been completed [11]. With regard to NICs, 10 Gbit/s adapters using ixgbe-driver or 1 Gbit/s adapters from Intel, RealTek and nVidia are supported. We found no documented restrictions for the usage of CPUs from different manufacturers [16].
Netmap uses standard system calls like `poll()` or `select()` for synchronization and other purposes [12]. While this reduces the probability of the system to crash, the performance may be reduced by additional context switches as described in section 2.4.
To ease the employment of netmap, a library that maps libpcap calls into netmap calls is provided. Furthermore, some functionalities like IP-forwarding are already implemented [16]. Notable examples of applications that added support for netmap are Click [28] and VALE [29].

4.2 PF_RING (DNA)

PF_RING is a network socket for capturing, filtering and analysing packets at high-speed, developed by ntop [14]. It is a kernel module that has the structure shown in figure 9. PF_RING uses the Linux NAPI drivers to poll packets from

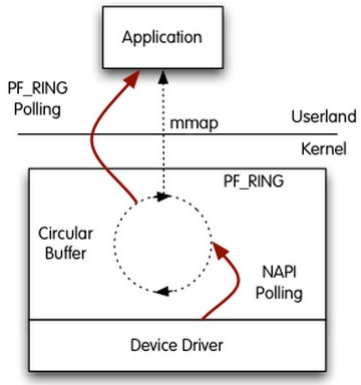


Figure 9: Vanilla PF_RING (image from [14])

the NIC and copy them into the PF_RING circular buffer. Using another polling mechanism the application in user-mode can read the packets from this ring. While two polling operations lead to increased CPU cycles spend per packet, the advantage lies in that "PF_RING can distribute packets to multiple rings, hence multiple applications, simultaneously" [14]. Furthermore, PF_RING implements memory pre-allocation for packet buffers, creation of parallel direct paths and memory mapping techniques [16].
A special feature comes with the PF_RING Direct NIC Access module [15]. This is a special driver to directly map NIC memory and registers to userspace, resulting in no copy operations besides the initial DMA transfer that is done by the NIC at packet arrival. Therefore, no interaction of the kernel takes place, apart from the polling operation of the application that gets the packets from the NIC memory and stores them into preallocated packet buffers in

doi: 10.2313/NET-2014-08-1_15

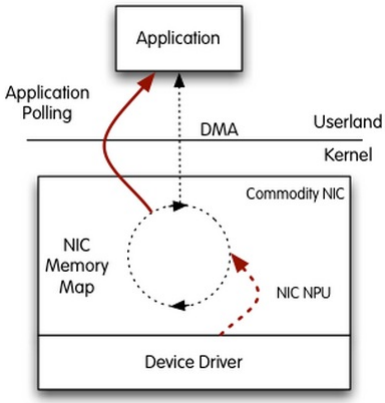


Figure 10: PF_RING Direct NIC Access driver (image from [14])

user space [13]. This behaviour is demonstrated in figure 10. While this allows for pure speed as the CPU only has to do one polling operation, it has the drawback that only one application at a time can access the DMA ring, thus, read packets from the ring. Furthermore, if the application misbehaves while accessing the memory and registers of the NIC, the whole system might crash [16].
PF_RING and all its modules are available for Linux kernels 2.6.32 and newer and support various 1 Gbit/s and 10 Gbit/s NICs from Intel. A wrapping of the PF_RING API that is similar to the libpcap library is also provided to allow flexibility and ease of use. On top of that, various sample programs for traffic monitoring, capturing and other purposes exist. One example is *pfsend*, a high-speed traffic generator.

4.3 Comparison: Intel DPDK, netmap and PF_RING DNA

The three presented frameworks show that different approaches to high speed packet processing can be considered. This begins with the decision whether to fully shift the processing to user space like it the Intel DPDK implements or use kernel modules instead. However, all frameworks tend to implement similar techniques in order to solve certain core problems which can be identified for example by having a closer look at the Linux network stack. A summary of those is given in table 1. All of the presented frameworks use their own structure to store the network packets. However, unlike the Linux network stack, they don't differentiate between packet data and several layers of headers, hence, reducing overhead caused by additional metadata. Packet buffers are always preallocated during start-up of the runtime and later on re-used, whereas the access to them is controlled using a ring structure. To reduce copy operations, memory mapping techniques are used, whereby PF_RING DNA brings it a step further in granting the application full access to the NIC's memory and registers. Unfortunately this comes at the cost of being prone to errors which may result in system crashes. As high reliability can be of great importance, one has to be particularly careful in the development of an application. Receive Side Scaling is supported by all three frameworks by creating parallel direct paths, allowing to distribute the load over all cores. One disadvantage that is difficult to get around, is the inability of PF_RING DNA

	DPDK	netmap	PF_RING DNA
Memory pre-allocation	✓	✓	✓
Custom network buffer structure	✓	✓	✓
Memory mapping	✓	✓	✓
Batch processing	✓	✓	✗
Parallel direct paths	✓	✓	✓
Not prone to architecture-related system crashes	✓	✓	✗
Use of standard system calls	✗	✓	✗

Table 1: Implemented techniques and features of different frameworks (table from [16], modified)

for batch processing of packets. If this feature is vital for an application to achieve high performance, one is advised to use another framework instead. The same holds true for netmap, as common system calls are used, that may lead to reduced performance caused by excessive context switching. One general problem of packet processing frameworks is that they are often limited to specific hardware and operating systems. Only if the required CPUs and NICs are present, specialised frameworks like the Intel DPDK or PF_RING (DNA) can be used, whereas netmap supports a broader range. Furthermore, all frameworks provide a different, yet libpcap-like, programming interface. Also, one has to keep in mind that only basic packet processing functions are delivered, whereas the user has to port any application to the chosen framework first. Example programs or libraries that provide additional functionalities are either provided by the developers themselves or have been created by communities over the past few years. Hence, one should look out for similar programs before writing an application from scratch.
As we found no work that tested the Intel DPDK, netmap and PF_RING DNA under same conditions, it is hard to compare the three frameworks regarding their achieved performance. However, tests from Intel using the Intel DPDK [6], [12] using netmap and [16] using PF_RING DNA (and several other packet processing frameworks) show that they are able to achieve the line rate of 10 Gbit/s even with 64 byte packets. Of course, this depends on the used hardware and can be different for other applications and use cases. To determine which framework is suited best for which application, generalized tests under equivalent conditions (for instance equal hardware) have to be made.

5. CONCLUSION

In this paper we described the problems of high-speed packet processing using modern commodity hardware. Using the Linux network stack as an example, several performance limiting factors have been identified. The basic functionalities of the Intel Dataplane Development Kit have been explained in detail. Furthermore, this framework has been compared with two other projects, netmap and PF_RING DNA, to illustrate some techniques that solve the described problems.
Commodity hardware is a valid alternative to packet processing using dedicated hardware as the software is always configurable to suit the current problem in the best possi-

doi: 10.2313/NET-2014-08-1_15

ble way. Several solutions have been created over the past few years to achieve the best performance that the hardware allows. While the presented frameworks differ in their respective implementations, they all try to resolve the core problems of existing, widely distributed packet processing solutions: per-packet allocation and deallocation, multiple memory copies, restrictions to parallel processing through locking mechanisms and context switching.
One has to keep in mind, that such frameworks are not designed to fulfil everyday general purpose networking tasks. Widely distributed and established solutions like the Linux network stack will retain the upper hand in those scenarios, as they support for instance a broad range of protocols. When it comes to high performance packet processing, the Intel DPDK, netmap and PF_RING DNA have the clear advantage. They are all capable of reaching the line rate of 10 Gbit/s for a specific task. The first applications like the DPDK vSwitch or VALE are becoming more and more popular as they function reliable at high speed.
However, packet processing frameworks like the Intel Dataplane Development Kit or PF_RING only support selected hard- and software. Furthermore, new problems arise for the end-user that has to write an application, as all frameworks provide different APIs. As some of those projects have dedicated communities like *dpdk.org*, the problem of porting applications to a specific framework can be eased significantly. Therefore, packet processing frameworks and commodity hardware are the way to go for the future, because hardware is constantly evolving, allowing for increased performance. The same trend must be maintained by the respective sophisticated software to become widely distributed and, thus, the de facto standard for packet processing.

6. REFERENCES

[1] C. Benvenuti: *Understanding Linux Network Internals*, O'Reilly Media, Inc., Volume 1, 2005

[2] R. Rosen: *Linux Kernel Networking: Implementation and Theory*, Apress, Volume 1, 2013

[3] M. Rio et al.: *A Map of the Networking Code in Linux Kernel 2.4.20*, in: Technical Report DataTAG-2004-1, March 2004

[4] W. Wu et al.: *The Performance Analysis of Linux Networking - Packet Receiving*, in: International Journal of Computer Communications, 30(5), pages 1044-1057, 2007

[5] *Intel DPDK: Data Plane Development Kit Project Page*, `http://www.dpdk.org`, last visited: June 2014

[6] Intel DPDK: *Packet Processing on Intel Architecture*, Presentation slides, 2012

[7] Intel DPDK: *Getting Started Guide*, January 2014

[8] Intel DPDK: *Programmers Guide*, January 2014

[9] Intel Open Source Technology Center: *Packet Processing*, `https://01.org/packet-processing`, last visited: July 2014

[10] Intel Open Source Technology Center: *Packet Processing - Intel DPDK vSwitch*, `https://01.org/packet-processing/intel%C2%AE-onp-servers`, last visited: July 2014

[11] *The netmap Project*, `http://info.iet.unipi.it/~luigi/netmap/`, last visited: July 2014

[12] L. Rizzo: *netmap:a novel framework for fast packet I/O*, in: Proceedings of the 2012 USENIX Annual Technical Conference, pages 101-112, 2012

[13] L. Deri: *Modern packet capture and analysis: Multi-core, multi-gigabit, and beyond*, in: the 11th IFIP/IEEE International Symposium on Integrated Network Management (IM), 2009

[14] ntop: *PF_RING Project Page*, `http://www.ntop.org/products/pf_ring/`, last visited 10.06.2014

[15] ntop: *PF_RING DNA Project Page*, `http://www.ntop.org/products/pf_ring/dna/`, last visited: June 2014

[16] J. García-Dorado et al.: *High-Performance Network Traffic Processing Systems Using Commodity Hardware*, in: Data Traffic Monitoring and Analysis, Springer Verlag, pages 3-27, 2013

[17] *Wikipedia: spin-lock*, `http://goo.gl/f1Dqhx`, last visited: June 2014

[18] Red Hat Enterprise Linux 6: *Performance Tuning Guide*, Chapter 8.6 Receive-Side Scaling (RSS), `http://goo.gl/qhGQYT`, last visited July 2014

[19] *Linux man page: socket*, `http://linux.die.net/man/7/socket`, last visited: July 2014

[20] lwn.net *A lockless ring-buffer*, `http://lwn.net/Articles/340400/`, last visited: June 2014

[21] G. Carle et al.: *Measurement and Simulation of High-Performance Packet Processing in Software Routers*, in: Proceedings of Leistungs-, Zuverlässigkeits- und Verlässlichkeitsbewertung von Kommunikationsnetzen und Verteilten Systemen, 7. GI/ITG-Workshop MMBnet 2013, Hamburg, Germany, September 2013

[22] M. Dobrescu at al.: *RouteBricks: Exploiting Parallelism To Scale Software Routers*, in: Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles, pages 15-28, October 2009

[23] R. Bolla et al.: *Linux Software Router: Data Plane Optimization and Performance Evaluation*, in: Journal of Networks, 2(3), pages 6-17, 2007

[24] S. Han et al.: *PacketShader: a GPU-accelerated software router*, in: ACM SIGCOMM Computer Communication Review, 40(4), pages 195-206, 2010

[25] F. Fusco: *High speed network traffic analysis with commodity multi-core systems*, in: Proceedings of ACM Internet Measurement Conference, pages 218-224, 2010

[26] N. Bonelli: *On multi-gigabit packet capturing with multi-core commodity hardware*, in: Proceedings of Passive and Active Measurement Conference, pages 64-73, 2012

[27] *Open vSwitch*, `http://openvswitch.org/`, last visited: June 2014

[28] R. Morris et al.: *The Click modular router*, in: ACM Transactions on Computer Systems (TOCS), 18(3), pages 263-297, 2000

[29] L. Rizzo and G. Lettieri: *VALE, a switched ethernet for virtual machines*, in: CoNEXT, pages 61-72, 2012.

Privacy Strategies in Smart Metering

Tobias Klenze
Supervisor: Benjamin Hof
Seminar Innovative Internettechnologien und Mobilkommunikation
Chair for Network Architectures and Services
Department of Informatics, Technische Universität München
Email: klenzet@in.tum.de

ABSTRACT
This paper discusses the threat to privacy that smart meters pose and offers solutions that protect user privacy without diminishing the provider's capabilities to use the data for legitimate purposes. Two use cases are considered: collection of aggregated real-time data by the provider for optimization, and monthly billing of customers. For the first, we present various approaches before proposing our own strategy for an aggregation protocol. For the second, we outline a billing protocol that preserves user privacy.

Keywords
Smart metering, Privacy, Data aggregation, Anonymization, Security, Homomorphic encryption.

1. INTRODUCTION
Smart meters currently undergo a deployment in many countries. While their advocates promise benefits to users and the grid infrastructure, the privacy implications of high-frequency data collected by smart meters has received little attention by the industry.

According to goals set by the European commission, 80 percent of alls EU households shall be equipped with smart meters by 2020 [1]. Advocates cite different reasons supporting wide-spread smart meter deployment. Automated billing eliminates the need for periodic inspections of meters by utility providers. It is also anticipated that integration of smart meters into a smart grid will facilitate fraud detection. Furthermore, electricity providers are interested in smart meters, because they enable complex tariffs, charging the customer more in peak consumption hours and less during hours with an abundancy of electricity. By combining smart meters with "smart appliances" that are sensitive to real-time electricity prices, it is possible to reduce the consumption during peak hours. For example, electric cars may charge up during the night, instead of busy hours during the day, thus reducing the capability requirements of the grid during peak hours, ultimately leading to fewer power plants being required. Awareness of current consumption on the user side may also lead to an overall reduction: studies show that 3-5% less energy is consumed when real-time meter readings are provided to consumers [8]. Finally, smart meter advocates state that real-time usage information will lead to a higher efficiency in managing grids and demand forecasting, facilitating decisions such as whether to start up a power plant and detecting leaks in pipelines as early as possible [7].

Whether or not these expectations of the capabilities of a "smart grid" (of which smart meters are a part) are justified, is beyond the scope of this paper. There have already been over three billion Euros of smart grid investment in the EU.[2] The fact of increasing deployment is reason enough to examine the implications of smart meters for consumers, particularly for their privacy. Current smart meters output usage data at a high-frequency, for example every 15 minutes, and report this usage to utility or grid providers. This is in contrast to old electromechanical meters, for which the provider only has low-frequency data, for instance on a monthly basis. As we will present in the next section, the higher frequency in usage reports to the provider has a drastic impact on the user's privacy.

In some countries, efficient energy usage and user privacy are debated as a dichotomy. Either the preference is given towards the benefits of smart meters and privacy detriments accepted as a necessary consequence, or the opposite holds, and smart meters are rejected altogether [7]. We will present protocols that protect the user's privacy while providing a high level of security.

Satisfying privacy and security at the same time is a difficult task, as the use case of billing with time-dependent tariffs illustrates. In this use case, monthly bills should be calculated from high-frequency usage data. On the one hand, the utility has an interest in computing the bill itself, so as to be secure against manipulation by the consumer. On the other hand, it must not gain access to fine-grained usage data for privacy reasons. Protocols that we are going to present in this paper use cryptographic mechanisms to ensure both privacy and security.

When designing a secure and privacy-preserving smart meter system, one also has to take device tampering into account. Attacks on smart meters are not merely theoretical. A 2011 study cites an FBI report, stating that as much as 10% of smart meters in one US state had been tampered with [7].

In this paper, we will present approaches that deal with privacy problems introduced by smart meters. The goal is to give an overview over the set of problems, while restricting the solutions presented to those that illustrate the different strategies in preserving the privacy of smart meter users.

We will start by presenting privacy problems in smart me-

doi: 10.2313/NET-2014-08-1_16

ters and then discuss at strategies that mitigate these issues while retaining the desired features of smart meters that smart meter advocates anticipate.

2. PROBLEM STATEMENT

We distinguish two different types of analysis that have implications for user privacy. The first is a statistical analysis that makes it possible to identify household appliances, such as a TV, a shower or a toaster. The second one is aimed at providing more abstract information, such as whether a person suffers from insomnia, or what religious group a resident belongs to. For the second type, it might be necessary to first identify individual appliances. Not all abstract questions about a household require such analysis however.

2.1 Identifying appliances

It has been shown over the last years, that smart meter data of sufficient quality can be used to identify single appliances with great certainty using statistics and machine learning. Existing approaches use high-frequency usage data, mainly of electricity consumption, to identify a number of household devices, such as televisions, washing machines, computers, refrigerators and even low-powered devices such as alarm clocks [11].

Figure 1 shows an example, where power consumption has been recorded over the period of one day. Alongside the consumption is annotated which appliances and which abstract information the authors of [9] have been able to identify using statistical methods and machine learning.

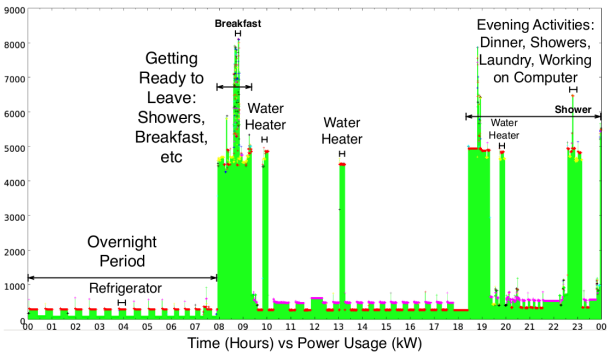


Figure 1: A sample identification of appliances using signatures. Source: [9]

2.2 Profiling on behavioral patterns

As the graph above demonstrates, using appliance signatures, it is possible to gain more abstract insights and do user profiling. For example, data on individual appliances can be utilized to answer questions such as: "Is the resident at home?" Imagining a utility provider asking this question about their employees on sick leaves, one can see why this data is sensitive and worthy of protection. After individual appliances have been identified, it is possible to answer these kinds of abstract questions [9].

It is difficult to tell which kind of data enables to such profiling. Some conclusions require accurate, high-frequency data, while others can be made with low-frequency and less accurate information. Profiling can not be completely prevented, but only limited. Abstract questions of the form "How many residents does a household have" or "Did all the residents go on vacation" can still be answered, even after applying the methods presented in this paper. For these questions, an analysis of very low-frequency data, maybe even as low-frequency as the monthly bill is sufficient. Evaluating how much profiling the data provided in a certain protocol permits is thus difficult. The more accurate, high-frequency data is available to the adversary, the easier profiling becomes. We will therefore aim to minimize accurate, high-frequency data availability to the utility and third parties.

3. SOLVING THE PRIVACY PROBLEMS

We will proceed to present solutions to the problems outlined in the last section. Two different use cases for high-frequency data will be distinguished and discussed separately.

3.1 Applications for smart meter data

In recent years, a lot of papers have discussed smart meter privacy problems. They are very diverse in the problem statement and the proposed solutions. For instance, some research how smart meter data sharing with third parties is possible in a privacy-preserving manner, and presuppose that the utility has access to sensitive information [6, 13]. This is contrasted by most papers which consider the relationship between the end user and the utility provider (and potentially a smart grid operator).

Leaving special cases aside, there turn out to be two common use cases for high-frequency smart meters data that have important privacy implications. The first is real-time data sharing with the utility and the grid provider. This data is used for management of the grids and plants, to have accurate and live information on utility usage, for demand forecasting, to detect problems and leaks. It has been noted that this use case is difficult to define, since there is no detailed specification on which data utility providers and grid operators require to achieve these goals [4]. Some providers try to collect as much data as possible, which is contrary to one of the most important principles in privacy design strategies, that data collection should be minimized [5]. Whether consumption data from individual households is actually necessary to achieve the goals outlined above, or whether the accuracy and high frequency is required, is doubtful. We will present approaches that contest these assumptions in order to allow for protocols that protect user privacy.

The second use case of high-frequency metering data is billing with non-constant tariffs. We will outline a scheme that allows for sophisticated tariffs, while preserving user privacy and keeping complex computations outside the smart meter's limited hardware.

The last point is an important consideration. Smart meters are usually low-end systems both in terms of computational power and bandwidth [7]. Solutions which require computationally expensive operations or different hardware than what currently-produced smart meters offer, are less likely to be deployed by providers. In addition, communication overhead should be kept to a minimum. It is due to these reasons that protocols for smart meters are usually de-

doi: 10.2313/NET-2014-08-1_16

signed to have minimal computations inside the trusted and tamper-resistant device itself. Instead, they only use the smart meter to do basic operations and outsource computationally expensive operations to other, perhaps untrusted devices.

3.2 Real-time data sharing

This section presents and discusses strategies for providers to capture real-time usage information without diminishing the user's privacy. We will first present two approaches that we regard not to be a sufficient solution to the problems outlined in the last section and then take a closer look at a more sophisticated protocol, which provides better protection of user privacy. Following our critique of this approach, we present our own modification of the protocol which aims at protecting users in a better way against maliciously acting smart meters.

3.2.1 Preprocessing

Preprocessing data is a strategy to hinder or at least to impede inferences that usage data permits. The idea is to obfuscate some of the information, making detailed analysis harder, without having an effect on the use of aggregated data. By preprocessing real-time usage data in certain ways, it is harder to identify appliances and thus it is also more difficult to come to certain abstract conclusions. For example, the data might still reveal that a person is in a household, but not whether she is working, or watching TV.

In [11], Reinhardt, Englert, and Christin discuss two different ways of data preprocessing: quantization and down-sampling. The first is restricting the possible values of reported power consumption to a multiple of a certain factor. The latter reduces the frequency with which readings are reported. For instance, one might employ both techniques and record usage only as a multiple of $q = 100$ *watts* (instead of a higher precision). Then, instead of reporting this consumption at a very high frequency, for instance every second, it is reported only every $t = 900$ *s*.

The authors evaluate both methods on their own and in combination. In their model, quantization is effective, whereas down-sampling has a smaller impact on appliance detection rate. Before applying any kind of preprocessing, their machine learning system is able to identify more than 90 percent of appliances in a sample. Applying quantization with $q = 180$ *watts* decreases accuracy to 58 percent, while applying down-sampling only decreases it to 74 percent with $t = 400$ *s*. Combining both techniques with said values still leaves the machine learning system able to identify 38 percent of all appliances.[1]

While making it more difficult for adversaries to draw conclusions given the usage data of a household, preprocessing is far from making it impossible altogether. Even with a detection rate of only 38 percent, there remain inferences that can be made from the preprocessed data, that problematic in terms of user privacy.

[1]It should be noted that [11, 9] started with a frequency of $1Hz$, which is higher than current smart meter's output frequency.

3.2.2 Multi-party protocols

Since preprocessing data is not sufficient to fully solve privacy problems with real-time smart meter data, we turn to a different approach. The assumption behind it is that providers do not require real-time data from each household. Instead, they can rely on aggregated data from multiple households. By obscuring the individual contributions of households to the aggregated value, their privacy is upheld.

A naive implementation of this concept is to employ a trusted third party (TTP), that aggregates each meter's readings and passes on the total to the provider. This proposal only shifts the privacy problems to the TTP. There are a number of authors who propose aggregation protocols without a TTP [7, 3, 4, 14]. For instance, [3] addresses the privacy problem by distributing data over different providers. The smart meter divides its consumption data into K different shares (which summed up, are the actual consumption), each of which is sent to a different provider. The idea is that a provider cannot infer information about an individual household with only a share of the actual consumption. However, each provider can aggregate the data and share that with other providers, thus enabling them to use high-frequency data without gaining insights on any individual household.

This approach is more promising than using a TTP, but it has a severe problem: Individual privacy is lost when an adversary learns the data shares sent to all K different providers. This could be due to collusions by the providers, but a perhaps more realistic scenario is when they are forced to hand over information about individual shares to a government agency. We therefore conclude that this strategy does not provide adequate protection in these adversarial models.

3.2.3 "No-Leakage" aggregation protocol

We will now present an aggregation protocol (called the "No-Leakage Protocol") that was proposed by Garcia and Jacobs [4]. We will then discuss drawbacks of the underlying privacy strategy and offer a modified version of the protocol in the next section.

The protocol gives the provider aggregated real-time consumption data of multiple households without revealing their individual shares. It uses homomorphic encryption in an asymmetric cryptography setting.

Homomorphic Encryption. An encryption scheme is homomorphic if it allows for sensible operations being carried out on ciphertext. For example, this could be an operation $\oplus$ on two encrypted values $enc(m_1)$, $enc(m_2)$, such that $enc(m_1) \oplus enc(m_2) = enc(m_1 + m_2)$. This encryption scheme is then called additively homomorphic.

A common additively homomorphic scheme is Paillier's cryptosystem, where:

$$dec(enc(m_1) \cdot enc(m_2) \mod n^2) = m_1 + m_2 \mod n$$

It is possible to multiply the encrypted value by a constant:

$$dec(enc(m)^k \mod n^2) = m \cdot k \mod n$$

doi: 10.2313/NET-2014-08-1_16

Setting and Infrastructure. The setting which we assume to work on consists of a local distribution station S and N households, each with a smart meter M_i connected to S. The goal is to provide near real-time usage information in aggregated form to the distribution station without revealing any individual contribution to the total consumption. The local station has a direct data connection to every household's smart meter. All data connection lines are assumed to secure the integrity of messages exchanged between two directly adjacent parties.

In the setup phase, each party (all M_i and S) creates a public/private key pair and we assume that there is a mechanism which distributes the public keys to all parties (e.g. by using a PKI).[2] It is assumed that the encryption scheme used is additively homomorphic, with $\oplus$ being multiplication (as in the example above).

The protocol may require a minimum number of participating smart meters. Since the security of the protocol is weakened with too few participants, the smart meters should not execute the data exchange if there are fewer than the minimum number of smart meters.

Protocol flow. After each measurement interval (for example, every 15 minutes), smart meters report on their household's consumption. Instead of sending their usage data directly to the local station S, they prepare N shares that are each sent to a different of the $N-1$ smart meters connected to the local station, except for one share, which remains at the meter. The shares are each encrypted using the recipient smart meter's public key (so that the local station, which has to relay the messages, does not learn the shares in plaintext). More formally, each smart meter M_i computes:

- Its own consumption m_i.
- Random numbers $a_{i1}, a_{i2}, \ldots a_{i(N-1)}$, and $a_{iN} \equiv m_i - \sum_{j=1}^{N-1} a_{ij} \ mod\ n$ (using a sufficiently large n).
- Encrypted values $y_{ij} = enc_{pk_j}(a_{ij})$ for each $1 \leq j \leq N$, $j \neq i$, where pk_j is j's public key exchanged during setup. a_{ii} is not encrypted and remains at the meter.

All the smart meters then send their y_{ij} values to S. S then computes for each j:

$$s_j := \prod_{i \neq j} y_{ij} = \prod_{i \neq j} enc_{pk_j}(a_{ij}) = enc_{pk_j}(\sum_{i \neq j} a_{ij})$$

Note that this equation holds due to the additive homomorphic encryption properties of the underlying cryptosystem.

S sends s_j to M_j, for all j.

[2]The establishment of trust is an important challenge, but cannot be resolved in this paper. For our purposes, we assume that each party receives the correct public key of each other party.

M_j decrypts s_j using his private key sk_j, yielding $\sum_{i \neq j} a_{ij}$. To this, M_j adds his own a_{jj}. M_j finally sends this value to S:

$$r_j := \sum_{i \neq j} a_{ij} + a_{jj} \ mod\ n$$

The addition of a_{jj} is essential, since otherwise S could exploit M_j's willingness to decrypt any ciphertext. If S probed M_j with individual shares (y_{ij}) instead of aggregated shares (s_j), M_j would return the plaintext shares (a_{ij}) to S. To avoid this, the value of r_j is obscured by the random value a_{jj}. As a final step of the protocol, S sums up all reported values:

$$total := \sum_j r_j = \sum_j \sum_{i \neq j} a_{ij} + a_{jj} \ mod\ n = \sum_i \sum_j a_{ji} \ mod\ n$$

This equals the total consumption of all households.

Proofs. Alongside the description of this protocol, [4] includes a proof of correctness and a proof of a "no leakage" property. For the former, it is merely noted that each a_{ij} is sent exactly once in a r_j message to S (by meter j) and that the sum of all shares a_{ij} over i and j is equal to the total consumption.

The "no leakage" property states that even if all but two meters are corrupted and act to reveal the uncorrupted meter's usage, no information about the latter meters' readings is leaked. This is an important result, since it means it is infeasible for colluding parties (e.g. S and a number of meters) to learn the consumption of other meters.

To accurately define this property, a "no leakage game" is defined. The proof relies on the underlying encryption scheme being IND-CPA secure (which is a formal definition of security in context of encryption). It shows that an adversary who wins the "no leakage game" can also win the IND-CPA-Game of the encryption scheme.[3] Presenting this proof is beyond the scope of this paper. Instead, we will discuss disadvantageous design choices of the protocol in the next paragraph and present a variation of the protocol addressing these problems in the next section.

Discussion. The "No-Leakage Protocol" is a protocol between a local station (operated by the provider) and individual household meters. It gives smart meters more control over the process of aggregation and limits the information that providers can learn. It therefore fits well the author's paradigm "Power to the Meter" which could be interpreted to be contrary to the status quo, which gives "Power to the Provider".

[3]The paper does not discuss the underlying encryption in detail, but mentions Paillier's cryptosystem as being IND-CPA secure.

doi: 10.2313/NET-2014-08-1_16

Taking power from the provider makes it more difficult for them to learn anything about a household's real-time consumption, but depending on the adversary model, it does not make it impossible. In the "Power to the Meter" setting proposed by the protocol's authors, meters are assumed to be trustworthy. In a scenario where the utility or grid operator is assumed to be the adversary trying to extract real-time usage data, this is a dubious assumption. Providers could influence the trusted computing base of smart meters, making all smart meters corrupted and thus violating the premise required for the security proof.

If all meters are corrupted, a naive way for them to leak usage information while following the protocol is to let each smart meter i set $a_{ij} = 0$ for all $i \neq j$ and $a_{ii} = m_i$. Then the actual consumption is revealed to the station S. To counter this attack, one might allow the user (i.e. household residents) to monitor the connection between their smart meter and S. The users would be able to detect such a simple attack. However, there are more sophisticated attacks, where detection is not possible. For example, a_{ij} for all i and j s.t. $i \neq j$ might be chosen from a pseudorandom sequence that is available to all the smart meters and the local station S. Each M_i uses these pseudorandom values, and sets a_{ii} in a way such that over all j, a_{ij} add up to m_i (as the protocol demands). To learn the consumption by meter j, S has to subtract the sum of all a_{ij} over i, s.t. $i \neq j$ from r_j (these a_{ij} values are known to S, since they belong to the pseudorandom sequence that S and all meters share). The user monitoring the message exchange cannot distinguish between a honest smart meter using random values and a malicious smart meter that uses pseudorandom values known to the utility to leak high-frequency data.

If smart meters are assumed to be untrusted, then they cannot be trusted with creating random shares. Therefore, we propose that the user provides the random values used in the computation. We will present such a protocol, under the paradigm "Power to the User" in the next section. The protocol aims at protecting privacy, even if all of the meters are corrupted.

It should be noted that in a setting where smart meters are assumed to be untrusted, their accuracy while recording consumption data is also in question. For example, the smart meter might over- or underreport actual usage during the real-time data aggregation protocol or the billing protocol. This however, can be detected, at least with some effort. For example, old electromechanical meters could be operated in sequence to the smart meter. Readings could then make any deviation of the smart meter apparent.[4] Such an ability of detecting a dishonest smart meter is not possible in the "No-Leakage" setting, since it is not possible to distinguish randomness from pseudorandomness using statistical methods.

3.2.4 User-controlled aggregation protocol

We propose modifications to the protocol, which lead to a new protocol under the "Power to the User" paradigm. The principal concepts are:

[4]The meter's own consumption has to be taken into account when comparing the meters.

1. Introduce the user U_i as a third party to meters and the local station.
2. Allow message exchanges between a smart meter and the local station only via the user.
3. Let the user and the corresponding smart meter each carry out the same computations as in the original protocol, with the user providing the random numbers and encryption parameters.
4. In particular, do so for message encryption: There is a shared key pair (spk, ssk) between the user and the meter. Both smart meter and user have the shared secret key ssk. The key pair is created by the user and then shared with the meter. The encryption scheme used is homomorphic.
5. Let the meter verify the correctness of the user's computations and let it use digital signatures to certify correctness and for integrity protection of the reported shares. This is based on a different key pair (pk, sk) for which the private key is available only to the meter and not the user.
6. Include counters in signatures to protect against replay attacks.

Figure 2 shows the full protocol flow. We do not display the protocol's initialization, where public keys are distributed to all parties and where all parties check whether enough meters participate in the protocol. Some parts of the protocol are simplified to reduce the complexity. For example, we did not include the fact that U_i will send parameters of the underlying encryption scheme to M_i, so that both parties get the same ciphertext.

The counters d_1, d_3 and d_3 are not formally defined. They provide protection against message replay attacks. Their values should not overlap (so that signatures of one step of the protocol can't be used for other steps). If there is a duplicate or missing d value, the protocol should be aborted (we rely on lower-level mechanism to ensure reliable transport). If signature verification fails at any step for any party, the protocol should be aborted as well.

Discussion. Our protocol makes the message flow between the meter and the station transparent to the user. It also aims at eliminating intentional leaks by meters, by letting the user provide random values. Making assertions on the protocol's properties (such as non-leakage) requires rigorous security proofs. Unfortunately, these are not as easy as in the original "No-Leakage" protocol. Introducing a third party and a variety of possible adversaries (meters, users, the local station or any combination) makes a formal analysis more difficult. The complexity of the protocol is also higher, as indicated by the increased number of steps and the signature checks required.

We will therefore only give a short account of what a security proof might show. In contrast to the original protocol, our version takes into account the case, in which all meters

doi: 10.2313/NET-2014-08-1_16

S		U_i		M_i
			$\xleftarrow{m_i}$	
		Pick $a_{i1}, \ldots a_{iN}$ For $j, j \neq i : y_{ij} := enc_{spk_j}(a_{ij})$		
			$\xrightarrow{a_{ij}, y_{ij}}$	
				For $j, j \neq i$: $y'_{ij} := enc_{spk_j}(a_{ij})$ Verify $y'_{ij} = y_{ij}$, $m_i = \sum_j a_{ij}$ $sig_{ij} := sign_{sk_i}(d_1, y_{ij})$
			$\xleftarrow{sig_{ij}, d_1}$	
	$\xleftarrow{y_{ij}, sig_{ij}, d_1}$			
Verify d_1 and sig_{ij} for all y_{ij} $s_i := \prod_{j \neq i} y_{ji} = enc_{spk_i}(\sum_{j \neq i} a_{ji})$ $sigs_i := sign_{sk_S}(d_2, s_i)$				
	$\xrightarrow{s_i, sigs_i, d_2}$			
			$\xrightarrow{s_i, sigs_i, d_2}$	
		Verify d_2 and $sigs_i$ for s_i $r'_i := dec_{ssk_i}(s_i) + a_{ii}$		Verify d_2 and $sigs_i$ for s_i $r_i := dec_{ssk_i}(s_i) + a_{ii}$ $sigr_i := sign_{sk_i}(d_3, r_i)$
			$\xleftarrow{r_i, sigr_i, d_3}$	
		Verify d_3, $r_i = r'_i$ and $sigr_i$ for r_i		
	$\xleftarrow{r_i, sigr_i, d_3}$			
Verify d_3 and $sigr_i$ for r_i $total := \sum_i r_i$				

Figure 2: Flow of the proposed user-controlled aggregation protocol

are malicious. We assume that meters record usage data accurately, since this is a property verifiable by the user (using a second meter, as discussed above).[5]

Most important for user privacy is that no consumption information leaks, even if meters and the local station collude. This should hold even if a number of users are part of the collusion. The protocol will not be able to protect the user if he is the only honest party. As with the original protocol, the utility can simply subtract all the other meters' readings from the total consumption to get the uncorrupted user's data.

Users should also not be able to distort the aggregated reading in any way, that is to manipulate the shares such that the total computed by S is not the sum of the individual readings. Integrity protection is introduced for this purpose.

If this protocol indeed has the properties described above, then it is an important improvement over the original protocol described in [4]. Not only is the user able to prevent his meter from leaking information, she can also use the real-time data herself, for her own analysis and purposes.

3.3 Billing

[5]Unnoticeably small modifications by the meter of its readings could be used to encode information into the usage reports, if they are not rounded to a lower precision. It might therefore be necessary to include into the protocol preprocessing (quantization) of usage data.

In contrast to real-time aggregation of data, it can be assumed that in billing, each party has a profound interest in manipulating the data, since it affects the final bill. The bill is in its simplest form a sum that is owed by the consumer to the provider. As with the aggregation protocol, it is necessary to provide protection against replay attacks.

In this section, we will sketch a protocol proposed by A. Rial and G. Danezis in [12]. They use homomorphic encryption, zero-knowledge proofs and commitment schemes, to build a secure and privacy-preserving billing system. In conformity with our approach for real-time aggregation systems, this protocol gives the user a great amount of control over the data.

It should be noted that a very simple protocol would create the bill entirely inside the smart meter's trusted computing platform. The bill then is sent over a trusted line to the provider. This has two disadvantages. First of all, it requires that the smart meter carries out all of the computations – which might include billing according to complex tariffs – all in a low-powered, tamper-resistant environment. Secondly, it gives great power to the meter and does not enable the user to verify the data herself, or to do her own computations using the high-frequency data. In contrast, the protocol proposed only requires adding digital signatures to the meter outputs. Everything else for computing the bill is handled by a computer (untrusted by the utility) under the user's control.

doi: 10.2313/NET-2014-08-1_16

System and design goals. Beside providing security to the provider, the protocol should also provide security to the user. This means, under the assumptions that the individual signed readings are correct, that the provider cannot wrongfully claim to be owed a different sum under a certain tariff than what is actually owed. The user's privacy is the second important design goal, which means that the provider should not learn any high-frequency data. Lastly, meters should not be required to carry out complex calculations, except signing high-frequency reading outputs. As with our real-time data protocol, they only have a data connection to the user, and not the power company or grid operator directly.

The following is the setting which we assume to be in place:

- Meter M outputs the consumption *cons* during a time interval and *other* as information associated with the reading (for example, the time).
- Υ is a publicly known tariff policy, i.e. a pricing function that takes as an input $(cons, other)$ and outputs a *price*. This function is signed by the provider P and sent by it to the user U.
- The tariff policy can be linear (fixed price for each consumption unit during a specific time) or cumulative (price for each consumption unit may depend on how much has been consumed).
- The sum on the bill is calculated by applying the pricing function during each reading and adding up the price of the individual readings. The total on the bill is calculated by the user and sent to the utility, alongside a proof that the total was computed using the meter's readings.
- Each party creates a key pair and distributes its public key to other parties.

Various cryptographic primitives are used in the protocol, which will be presented.

Commitment schemes. A commitment scheme is a cryptographic scheme, where a party can commit to a certain value, without immediately revealing the value. It can later open the commitment by revealing the committed value and the commitment's unique opening. The distinctive feature of commitments is that it's not possible to find a two value/opening pairs that yields the same commitment. After a commitment is opened, all other parties can be assured that the value used for the opening was the value that was committed to.

A classical example for a useful application is the rock, paper, scissors game. Using commitment, each party can commit to a certain value and reveal the commitment's opening along with the committed values only after all commitments are public.

The protocol discussed here uses a homomorphic commitment scheme.

Zero-knowledge proofs. A proof of knowledge of some value that does not reveal the value itself, is called a zero-knowledge proof. It is a protocol between a prover and a verifier. Proofs can either be interactive, requiring message exchanges between both parties (such that the verifier can send challenges to the prover), or non-interactive, consisting of a single message by the prover. The protocol presented here uses non-interactive zero-knowledge proofs.

Protocol flow. For each reading tuple $(cons, other)$, the meter M creates a commitment to *cons* and a commitment *other*. It then creates a digital signature, using its own private key, over $(d, com(cons), com(other))$ (where d is a counter) and sends this signature along with the value d, the commitments and their the openings *cons* and *other* to the user.

The billing protocol is initiated at the end of the billing period to create a final bill from the individual $(cons, other)$ readings and the pricing policy Υ. The entire calculation of the bill, and proof of its correctness, happens outside the smart meter, i.e. on the user's home computer. After the bill has been transmitted to the utility alongside a proof of correctness, the utility verifies the bill and charges the customer with the total on the bill.

For each tuple $(cons, other)$ by the meter, the user creates a commitment to the price $price := \Upsilon(cons, other)$. It then creates a non-interactive zero-knowledge proof π of: (1) knowledge of the signed commitment's opening. (2) knowledge of the opening for the committed price, (3) knowledge of a digital signature by the provider P on $(cons, other, price)$ with which the provider certified that $\Upsilon(cons, other) = price$.

Next, the user uses the homomorphic property of the commitment scheme to aggregate the price commitment, producing a commitment $com(total)$. The user transfers both the commitment and the opening to this total sum $com(total)$ and $(total)$ to the utility. It also includes individual price commitments and the zero-knowledge proofs π. The entire message is signed by the user.

The provider verifies the user's signature on the message, the meters signature on $(d, com(cons), com(other))$ tuples and the proofs π. Afterwards it uses the homomorphic operation on the *price* commitments to get a commitment for the total price. It compares this to $com(total)$ and verifies if the user-reported *total* and opening opens the aggregated *price* commitment. Finally, via the counter d it is verified, if the user did include exactly the readings for the billing period.

The protocol requires that the user can be asked by the provider to give up some $(cons, other)$ tuples, by providing the openings to the committed values.

Outlook. The construction used by Rial and Danezis is too complex to fully explain in this paper. The same holds for the consideration of different tariff structures and the security proofs. However, the sketched protocol flow above outlines for which part of the protocol the cryptographic

doi: 10.2313/NET-2014-08-1_16

primitives presented above are required. It also shows, that while difficult, creating privacy-preserving billing protocols that allow for tariff computations outside smart meters is feasible.

4. CONCLUSIONS

In this paper, we have presented and discussed several solutions to the challenges brought up by smart meter technology. While current smart meters deployments largely ignore risks for the user' privacy, the results above show that these threats are substantial. We have argued that the problems are not of the form of an unresolvable conflict between privacy and the benefits that smart meters are claimed to introduce. To the contrary, over the past years a number of researchers have presented sophisticated approaches to deal with problematic high-frequency data. Even though providers are reluctant to share their requirements for real-time data collection, researchers are able to make educated guesses about which data is actually needed.

For the case of real-time data collection, we have seen three different proposals. While preprocessing helps to reduce the severity of the issues at hand, it does not fundamentally resolve them. As for the second approach, we argued that a protocol requiring that providers do not collude in order for privacy protections to be effective, provides little protection when adveraries are government agencies. The "No-Leakage" protocol presented is a promising approach to protect privacy while enabling real-time data collection. Instead of distributing the control over multiple providers, the control is given to the smart meters. Thus, in order for information to leak, almost all meters have to be corrupted.

This seems like a satisfying result, until one considers the influence that providers exert over the smart meter deployment. Attacks on privacy by providers who roll out corrupted (i.e. backdoored) smart meters to their customers form an adversarial model overlooked in the protocol's design. In such a scenario, the data protection mechanism fails. Worse yet, the user is unable to detect attacks. Without inspection of the software running on the tamper-resistant device, the user cannot decide whether the smart meter leaks data or honestly follows the protocol. This holds even if she is able to read all messages exchanged between the smart meter and the provider.

To protect the user against such attacks, we proposed a protocol similar to "No-Leakage", with the difference, that the user is in control. She provides the randomness used in the encryption key and the random shares. To prevent misuse, the smart meter verifies the correctness of the user's calculations and protects their integrity with a digital signature.

The protocol aims at ensuring privacy even if smart meters behave maliciously. However, more research is required to subject the protocol to a thourough security analysis and to make formally proven statements on its properties.

A further topic of research is the distribution of public keys to all parties. A public-key infrastructure with certificate authorities scales well, but has problems: If certificate authorities are run or controlled by providers, then trust cannot be established, since they are assumed to be possible adversaries. The users for one substation live in close proximity, such that alternatively a personal key exchange and a web-of-trust model might be feasible.

Lastly, we looked at the billing use case. There, security and privacy were in conflict in particular, since each party has an interest in manipulation of the bill. While it was only possible to sketch one approach in the restricted scope of this paper, it became apparent that complex tariffs can be implemented in a privacy-preserving manner.

The architectures both of the billing protocol and our aggregation protocol follow the "Power to the User" paradigm. We argued that in order for customers to have confidence in the protection of their privacy, it does not suffice to transfer control from the providers to the meters. The user must not only be able to read the message exchange, but act as an active participant in the protocols. Although further research is required to refine, evaluate and formally analyze the protocols presented in this paper, they are promising approaches for secure and privacy-preserving smart meters systems.

5. REFERENCES

[1] Directive 2009/72/EC of the European Parliament and of the Council, 13 July 2009.

[2] Smart Grid Projects Outlook, 2014. `http://ses.jrc.ec.europa.eu/smart-grids-observatory`

[3] C. Barcellona, P. Cassara, G. Di Bella, J. Golic, and I. Tinnirello. Multi-party metering: An architecture for privacy-preserving profiling schemes. In *Sustainable Internet and ICT for Sustainability (SustainIT), 2013*, pages 1–6, Oct 2013.

[4] F. Garcia and B. Jacobs. Privacy-friendly energy-metering via homomorphic encryption. In J. Cuellar, J. Lopez, G. Barthe, and A. Pretschner, editors, *Security and Trust Management*, volume 6710 of *Lecture Notes in Computer Science*, pages 226–238. Springer Berlin Heidelberg, 2011.

[5] J.-H. Hoepman. Privacy design strategies. *CoRR*, abs/1210.6621, 2012.

[6] P. Kumari, F. Kelbert, and E. Pretschner. Data protection in heterogeneous distributed systems: A smart meter example, October 2011.

[7] K. Kursawe, G. Danezis, and M. Kohlweiss. Privacy-friendly aggregation for the smart-grid. In *Proceedings of the 11th International Conference on Privacy Enhancing Technologies*, PETS'11, pages 175–191, Berlin, Heidelberg, 2011. Springer-Verlag.

[8] C. McKerracher and J. Torriti. Energy consumption feedback in perspective: integrating australian data to meta-analyses on in home displays. *Energy Efficiency*, 6(2):387–405, May 2013.

[9] A. Molina-Markham, P. Shenoy, K. Fu, E. Cecchet, and D. Irwin. Private memoirs of a smart meter. In *Proceedings of the 2Nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, BuildSys '10, pages 61–66, New York, NY, USA, 2010. ACM.

[10] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In J. Stern, editor, *Advances in Cryptology – EUROCRYPT 1999*,

volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer Berlin Heidelberg, 1999.

[11] A. Reinhardt, F. Englert, and D. Christin. Enhancing user privacy by preprocessing distributed smart meter data. In *Sustainable Internet and ICT for Sustainability (SustainIT), 2013*, pages 1–7, Oct 2013.

[12] A. Rial and G. Danezis. Privacy-preserving smart metering. In *Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society*, WPES '11, pages 49–60, New York, NY, USA, 2011. ACM.

[13] L. Sankar, S. Kar, R. Tandon, and H. V. Poor. Competitive privacy in the smart grid: An information-theoretic approach. *CoRR*, abs/1108.2237, 2011.

[14] C. Thoma, T. Cui, and F. Franchetti. Secure multiparty computation based privacy preserving smart metering system. In *North American Power Symposium (NAPS), 2012*, pages 1–6, Sept 2012.

doi: 10.2313/NET-2014-08-1_16

Der Wert von Privatsphäre bei Websuchen und Personalisierung von Websuchen

Sebastian Vogl
Betreuer: Heiko Niedermayer
Seminar: Innovative Internettechnologien und Mobilkommunikation SS2014
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: s.vogl@tum.de

KURZFASSUNG

In dieser Arbeit wird der Wert von Privatsphäre bei Websuchen unter Berücksichtigung von Personalisierung besprochen. Zu Beginn wird eine kurze Einführung in Anonymisierung und Personalisierung gegeben. Diese beinhaltet jeweils verschiedene Arten und deren Voraussetzungen. Danach wird auf das Hauptthema „Was ist der Wert von Privatsphäre bei Websuchen?“ eingegangen. Dazu werden ein Experiment und eine Umfrage herangezogen. Für das Experiment sollten Probanden verschiedene Websuchen ausführen. Dabei wurden deren Verhalten und Privatsphäre Einstellungen beobachtet und ausgewertet. Im weiteren Verlauf der Arbeit werden der Ablauf des Experiments und dessen Ergebnisse aufgezeigt. Nachfolgend wird der Aufbau der erstellten Umfrage erklärt und zur Bewertung der Ergebnisse des Experiments herangezogen.

Schlüsselworte

Websuche, Suchmaschine, Anonymisierung, Personalisierung, Profile, Privatsphäre, Datenschutz, Arten der Personalisierung, Wert von Datenschutz, Personalisierte Suche

1. EINLEITUNG

„Wie wichtig ist der heutigen Gesellschaft Ihre Privatsphäre?“ beziehungsweise „Wie wichtig ist mir meine Privatsphäre?“. Durch das Preisgeben persönlicher Angaben und eigener Interessen wird der Umgang auf Webseiten sehr vereinfacht. Jedoch den Zugriff dieser Daten wieder einzuschränken, oder sie gar zu löschen ist weitaus schwieriger als sie vorher zu erstellen. Ein Beispiel zu diesem Thema ist „Das Recht auf Vergessen“, wie es in den Medien genannt wird. Dabei wird Google nach dem Verlorenen Präzedenzfall [4] am Europäischen Gerichtshof dazu veranlasst, innerhalb der EU die Möglichkeit zur Löschung von eigenen Suchanfragenergebnissen zu geben. Zwar ist die grundsätzliche Idee nachvollziehbar, aber man kann im Moment davon ausgehen, dass, meiner Meinung nach, dieses Recht auf längere Zeit nur von wenigen genutzt werden wird. Zum einen liegt dies daran, dass der Vorgang aufwendig durchzuführen ist, indem man ein Formular ausfüllen und jede einzelne Anfrage explizit benennen muss, zum Anderen gilt dieses Recht nur für Google-Domänen in der Europäischen Union. [5] Die Hauptwebseite google.com ist davon nicht betroffen, da diese nicht innerhalb der EU betrieben wird. Weiter kommt hinzu, dass eine Kopie des Personalausweises der Privatperson hochgeladen werden muss, welche eine eindeutige Identifizierung durch Google ermöglichen würde. Dies wird von mehreren Seiten bemängelt. Ein Beispiel dazu findet sich unter [14]. Somit kommen folgende Überlegungen auf: „Was ist uns eigentlich die Privatsphäre bei Websuchen wert?“ und „Wie sieht Personalisierung heute aus?“

2. ANONYMISIERUNG UND PERSONALISIERUNG

Nach §3 Abs. 6 Bundesdatenschutzgesetzt [13] ist Anonymisierung definiert als „[…] das Verändern personenbezogener Daten derart, dass die Einzelangaben über persönliche oder sachliche Verhältnisse nicht mehr oder nur mit einem unverhältnismäßig großen Aufwand an Zeit, Kosten und Arbeitskraft einer bestimmten oder bestimmbaren natürlichen Person zugeordnet werden können.“. Somit zusammengefasst, das Verändern von Daten, dass sie nicht mehr einer Person zugeordnet werden können. Im Gegensatz dazu ist Personalisierung definiert als das Erheben und Verarbeiten von Daten einer Person, dabei ist das „Erheben […] das Beschaffen von Daten über den Betroffenen (Person).“ und das Verarbeiten „[…] das Speichern, Verändern, Übermitteln, Sperren und Löschen personenbezogener Daten.“

2.1 Möglichkeiten der Anonymisierung

Informationsbeschaffung war für die Menschen schon immer wichtig, doch nicht jedes Thema, das man im Internet anspricht, möchte man in sein öffentliches Profil mit protokolliert haben. Vor allem, wenn diese Protokolle eine Person, dessen Umfeld mit Freunden, Verwanden, dessen Kaufverhalten und Interessen bis ins kleinste Detail miteinschließen. Wie viel man mit solchen Datensätzen anfangen kann und, wie man diese wieder Vermarktet sieht man zum Beispiel an der Google Trend-Funktion [17], bei der die meist gesuchten Begriffe aufgelistet und Trend-Tendenzen ausgewertet werden. Deswegen wird gerne auch auf die Möglichkeit der Anonymisierung zurückgegriffen.

Es gibt verschiedene Arten seine Identität im Internet oder bei Websuchen zu verschleiern. Zum einen gibt es die Möglichkeit falsche Angaben (z.B. Alter, Geschlecht, Wohnort) bei der Anmeldung zu machen. Dies ist für den Nutzer die einfachste Möglichkeit von sich abzulenken. Andererseits hilft dies nicht

doi: 10.2313/NET-2014-08-1_17

bei dem Problem der Personalisierung, da bisher immer noch eine, wenn auch nur virtuelle, Person vorhanden ist. Auch bei der Verwendung von verschiedenen „Pseudo“-Profilen, besteht die Möglichkeit, dass mittels Verhaltensmuster-Analysen eine Personalisierung durchgeführt wird. Deswegen werden oft weitere externe Dienste hinzugezogen (z.B. Tor [6], JonDonym [7] und verschiedene Proxy oder VPN Anbieter), die nicht nur den eigenen Zugangsstandort, sondern auch die Identität einer jeweiligen Person durch eine öffentliche, von Mehreren genutzte, Identität ersetzen. Somit kann zum Beispiel im Falle von Websuchen, keine Personalisierung nach Interessen durchgeführt werden, da über die gleiche IP-Adresse unterschiedliche Nutzer mit ihren jeweiligen Interessen auf die Websuchmaschinen gleichzeitig zugreifen. Bei der Nutzung von externen Diensten besteht dennoch weiterhin die Gefahr der Datenspeicherung. Denn diese Dienste müssen, wenn auch nur temporär, die korrekten Daten der zu verschleiernden Personen abspeichern, damit der Netzwerkverkehr an die richtigen Personen weitergeleitet werden kann. Eine weitere Option um die Identifizierung von Personen aufgrund dieser Datenspeicherung zu vermeiden wäre die Nutzung von öffentlichen Internetzugängen (z.B. die Nutzung öffentlicher Hotspots an Bahnhöfen oder Flughäfen). Außerdem sollte man beachten, sobald man persönliche Daten, auch über eine anonymisierte Verbindung, versendet, kann man schnell wieder enttarnt werden. Eine letzte Möglichkeit, die verwendet wird und in dieser Arbeit genannt werden soll, ist das Verwenden einer widerrechtlichen erlangten Identität (z.B. das Nutzen eines privaten WLan- Anschlusses einer fremden Person). So wird zwar die eigene Identität verschleiert, aber dieses Vorgehen kann strafrechtliche Konsequenzen mit sich ziehen.

Da dies nicht Thema dieser Arbeit ist, wird nicht weiter darauf eingegangen. Sofern weiteres Interesse besteht, wird auf [15] und [16] verwiesen.

2.2 Arten der Personalisierung

Genauso wie es verschiedene Arten der Anonymisierung gibt, existieren auch verschiedene Arten der Personalisierung. Die folgenden Arten werden als Grundlage für Websuchen verwendet.

Die erste Art wird als „Relevance Feedback and Query Modification“ bezeichnet. Zwar dienen die Informationen davon nicht direkt der Personalisierung, aber im Datensammlungs-prozess ist es eine wichtige Grundlage. Das eigentliche Ziel hierbei ist ein besseres Suchergebnis zu finden, das auf Basis der eingegeben Informationen und bestimmten Ergebnis-gewichtungen erstellt wird. Dabei wird die ursprüngliche Suchanfrage intern durch andere Ergebnisse und Daten (z.B. Standortrelevanz) angepasst. Man kann das Vorgehen mit einer Personalisierung einer einzelnen Suchanfrage vergleichen.

Die zweite Art basiert auf der Analyse des Inhalts der Webseiten. Dabei werden Inhalte von verschiedenen Webseiten verglichen und auf Basis der Relevanz und den bisherigen Gewohnheiten beziehungsweise Interessen des Nutzers angepasst ausgegeben. Das Profil des Nutzers wurde bei dieser Methode bereits angelegt. Im Gegensatz zum vorher genannten „Relevance-System“ wird hier ein größerer Raum betrachtet als eine einzelne Suchanfrage. Sobald ein Nutzer eine Suchanfrage stellt, wird mithilfe des „Relevance-Systems“ eine Sortierung von Suchergebnissen zurückgegeben. Diese wird jedoch nochmal durch das zuvor erstellte Profil des Nutzers neu angeordnet.

Die bisherigen Arten sind relativ statisch und arbeiten auf einer bestimmten Vorgabe. Da es sehr aufwendig ist, ein gutes Profil zu erstellen, sollte dies das Analysesystem dynamisch selbst durchführen können. Dies ist nun Aufgabe der „Recommender Systeme“. Hierbei wird mithilfe von Algorithmen, die als Datenbasis Langzeitinformationen nutzen, die Analyse von Inhalten der Webseiten weiter an den Nutzer angepasst. Da dieses gesamte Konstrukt auf dem Relevance System basiert und dieses mit verschieden passenden Anfragen erweitert, um das Ergebnis zu strukturieren, muss das Recommender System abschätzen können, welche Suchanfragen der Nutzer an dessen Stelle geben würde.

Im Gegensatz zu den drei vorher genannten Systemen gibt es auch ein System, das auf der Analyse der Links basiert, und somit „Link-Analysis“ genannt wird. Bei diesem Ansatz, wird die Topologie des Internets zur Relevanzbestimmung genutzt. So muss nicht der Inhalt selbst auf Relevanz überprüft werden, sondern die Verlinkungen der Webseiten werden analysiert. Beim sogenannten topic-sensitive-PageRank-Algorithmus werden, basierend auf den Verlinkungen unterschiedlicher Webseiten und der Thematik selbst, verschiedene Kategorien und eine Wertung für Webseiten erstellt. Der personalisierte topic-sensitive-PageRank -Algorithmus ist eine Erweiterung des allgemeinen topic-sensitive-PageRank. Hierbei werden Nutzer Profile beziehungsweise deren Webseiten-Vorlieben in die Bewertung der Webseiten mit einbezogen.

Zuletzt soll noch ein weiteres System genannt werden, welches nicht nur auf einen Nutzer fixiert ist. Beim sogenannten „Social Search Engine“ wird das Verhalten von verschiedenen Nutzergruppen in die Wertung für Webseiten mit einbezogen. Dieser Ansatz entstand durch die Beobachtung der Gesellschaft, da zum Beispiel Bücher- oder Filmkäufe meist aufgrund von Empfehlungen Anderer durchgeführt werden. Die „Social-Search-Engine“ wird zu den genannten Arten verhältnismäßig öfter bei den „Recommender Systems“ verwendet, indem man Nutzer, die ein ähnliches Profil besitzen zusammen betrachtet. Die Seitenwertung wird damit unter dem Leitgedanken „Das Kollektiv hat (meistens) recht“ ausgeführt.

Schlussendlich kann gesagt werden, dass heutige Suchmaschinen auf einer Mischung der genannten Arten basieren. Außerdem fällt auf, dass für ein gutes Suchergebnis ein gutes Profil des Nutzers benötig wird, welches durch die Verwendung von dynamischen Systemen immer weiter verbessert wird und somit eine kontinuierliche Analyse des Nutzers voraussetzt. In [2] findet man einen tieferen Einblick in diese Thematik.

2.3 Voraussetzungen der Personalisierung

Um eine Personalisierung von Nutzern durchzuführen, reicht es nicht aus, verschiedene Daten nur in einer großen Liste zu speichern. Dazu muss man sich verschiedene Fragen stellen, die speziell für die Art und deren Ziel zu beantworten sind. Da für jedes Verwendungsgebiet jeweils andere Anforderungen vorhanden sind, werden hier nur Fragen angeführt, die im Fall einer Suchmaschinenpersonalisierung zu beantwortet sind. Im Folgenden wird hauptsächlich auf das Thema Personalisierung

doi: 10.2313/NET-2014-08-1_17

von Websuchen eingegangen und erklärt, wie diese optimal verwendet wird.

Angefangen wird mit der Frage „Wie beschafft man sich die Daten?“, beziehungsweise „Welche Methode soll man zu Speicherung verwenden?“ Hierbei wird zwischen direkter und indirekter Datenbeschaffung unterschieden. Da es für den Nutzer unangenehm ist, bei jeder Systemverwendung ein direktes Feedback zu geben, wird meist die indirekte Form verwendet. Hierbei wird das Verhalten von Nutzer und System beobachtet und teilweise noch mit der Möglichkeit der optionalen direkten Feedbackgabe erweitert. Für die Websuche bietet sich eine indirekte Datensammlung sehr gut an, da eine nahtlose Kommunikation zwischen Nutzer und System vorhanden ist und Suchanfrage eine gute Basis zur Personalisierung bieten.

Die nächste Frage ist „Wie oder wo sollen bereits erstellte oder noch zu erstellende Profile gespeichert werden?“ Auch hier gibt es zwei Möglichkeiten, die serverseitige und die clientseitige Speicherung. Die Vorteile der clientseitigen Datenspeicherung liegen in der erhöhten Privatsphäre und Sicherheit des Nutzers. So verbleiben private Daten auf dem Nutzersystem und können hier auch jederzeit gelöscht werden. Die serverseitige Datenspeicherung würde aber die Auswertung von Verhaltensmustern der Nutzer besser unterstützen, außerdem spielt das verwendete Clientsystem (z.B. verschiedene Arten von Betriebssystemen) keine Rolle. Andererseits erzeugt eine serverseitige Nutzung eine höhere Auslastung des Servers. Somit sollte eine Lösung gewählt werden, die beide Seiten, Client wie auch Server, optimal ausnutzt. Hierbei ist es zu beachten, wie oft ein Austausch von Informationen der beiden Seiten stattfindet. Eine durchgehende Verbindung ist dabei empfehlenswert, da Informationen unmittelbar auf beiden Seiten synchronisiert und somit aktuell gehalten werden können.

Eine weitere wichtige Frage ist „Soll sich das System über die Dauer der Verwendung an den Nutzer anpassen?“ Bei Websuchen ist diese Frage mit einem eindeutigen „Ja“ zu beantworten, da Websuchen zumeist auf dem Themen basiert, die für eine Person in diesem Zeitraum interessant sind. Da sich Interessen aber schnell ändern können, sollte die Möglichkeit der Anpassbarkeit gegeben sein. Empfehlenswert ist es, dem Nutzer selbst eine Anpassung des Systems zu erlauben. Diese Form findet man zum Beispiel oft bei werbefinanzierten Streaming-Portalen, die eine Grundeinstellung der Werbeinteressen ermöglichen, aber auch Suchmaschinenbetreiber stellen individuelle Konfigurationsmöglichkeiten bereit.

Wenn man sich überlegt, wie man die Profile an Nutzer anpasst, sollte auch die Frage „Wie werden die Profile aufgebaut und was sollen sie enthalten?“ gestellt werden. Um diese aber zu beantworten, sollte die Art und Methode der Personalisierung beziehungsweise deren Algorithmus festgelegt sein. Dabei kann man entweder auf verschiedene Drittanbieter zurückgreifen, an die nur die reinen Informationen gesendet werden, welche dann ausgewertet werden, oder man verwendet ein eigenes System, die die Relevanz der Suchergebnisse mit berücksichtigt und bewertet. Da die Entwicklung von korrekten und vor allem schnellen Algorithmen mit viel Aufwand verbunden ist, wird ein solcher Algorithmus entweder nicht oder nur mit hohen Lizenzgebühren veröffentlicht. Aus der Struktur und der Funktionsweise ergibt sich eine Vorgabe für den Aufbau der Profile, zum Beispiel wie verschiedene Themen gewichtet und wie die Informationen gespeichert werden (Bsp.: Liste, Tabelle, Cluster).

Letztendlich muss man sich noch überlegen: „Wie sieht die Nutzerschnittstelle aus?“ Dabei ist vor allem zu beachten, welche Arten von Clients verwendet werden. So verhalten sich Desktop-Webbrowser anders als mobile Webbrowser. Außerdem kann eine spezielle Nutzer-Applikation auf der Client Seite vorliegen, die ein bestimmtes Format vorrausetzen, zum Beispiel eine beschränkte Zeichenmenge, aufgrund von Ressourcenmangel.

Unter [2] werden die hier genannten Themen gezielter ausgeführt.

3. EXPERIMENT UND UMFRAGE

In den folgenden Kapiteln werden ein Experiment und eine Umfrage jeweils mit deren Ergebnissen zum Thema „Der Wert von Datenschutz bei Websuchen“ vorgestellt und verglichen.

3.1 Vorbereitung und Ablauf des Experiments

3.1.1 Durchführung

Das Ziel des Experiments war es herauszufinden, wie viel den Nutzern einer Suchmaschine die Privatsphäre wert sei. Dazu wurden zu verschiedenen Zeitpunkten am University College London (UCL) 189 Testpersonen eingeladen. Diese wurden vorher innerhalb des Universitätsbereichs angesprochen, ob sie an einem Test einer neuen Suchmaschine, welche „Find Fever“ genannt wurde, teilnehmen würden. Die Absicht das Verhalten und Sucheinstellungen zu beobachten wurde nicht explizit genannt, aber im Verlauf des Experiments wurde gefragt, welche Einstellung die jeweilige Person zum Thema Datenschutz hat.

Nachdem sich die Testpersonen registriert und angemeldet hatten, sollten sie verschiedene Fragen beantworten, wobei sie das Internet frei verwenden konnten, jedoch alle Suchmaschinenanfragen auf die „Find Fever“-Seite verlinkt wurden. Es wurden verschiedene Themen behandelt, welche zum Teil einen allgemeinen Charakter hatten, zum anderen Teil sensiblere Informationen abfragten. Folgend ein kurzer Ausschnitt aus dem Fragenkatalog [1]:

- „What is the population size of Little Shelford, England? “
- „What is the address of the tourist information in York, England? “
- „Where can you buy lingerie in Chelsea? “
- „What is the maximum penalty for attempted sexual intercourse with girl under 13? “
- „Which airport is closest to you? “
- „What is the proportion of water in human faces? “

Das Experiment selbst war in zwei Teile gegliedert. Im ersten Durchlauf waren die in Tabelle 1 gegebenen Einstellungs-möglichkeiten kostenlos verfügbar. Im zweiten Teil konnte man, wie in Tabelle 1 gezeigt, Credits für bestimmte Einstellungen ausgeben und für andere zurückgewinnen. Die vergebenen Credits wurden am Ende mit der gegebenen Entlohnung der

doi: 10.2313/NET-2014-08-1_17

Teilnahme an dem Experiment abgeglichen. (Ein Credit war ungefähr 0,30€ wert.) [1]

Tabelle 1. Einstellungsmöglichkeiten und deren Kosten beziehungsweise Gewinne während des Experiments

Einstellung	Credits
Speichere meinen Suchverlauf nicht.	-2
Entferne Werbung auf der Ergebnisseite.	-2
Gib meine Suchanfragen nicht an Dritte weiter.	-2
Verbessere das Suchergebnis.	-1
Hebe die Suchanfrage in den Ergebnissen hervor.	-1
Kopple die Suchergebnisse mit meinem Standort.	kostenlos
Verwende die sichere Suche Funktion.	kostenlos
Veröffentliche meine Suche auf Twitter.	+2
Speichere, auf welche Ergebnisse ich klicke.	+2

3.1.2 Aufgestellte Hypothesen

Vor Beginn des Experimentes wurden andere Versuche betrachtet. Die Versuche sind unter [8] und [9] beschrieben. Dabei fand man heraus, dass Privatsphäre schon ein wichtiger Teil in Online-Shops ist. So sind Kunden bereit, mehr Geld für Waren auszugeben, wenn ein sensiblerer Umgang mit ihren Daten gegeben ist. Dennoch gibt es Abstufungen, wie viel mehr ein Produkt kosten darf und auf welche Art in die Privatsphäre eingedrungen wird. Somit zeigten die Ergebnisse, dass, sobald der Rabatt verlockend genug war, auf in die Privatsphäre eingreifende Webseiten gewechselt wurde. Des Weiteren wurde festgestellt, dass zum Beispiel kaum mehr Geld ausgegeben wurde, wenn es um das Versenden von Werbung an die E-Mailadressen der Nutzer ging. Siehe hierzu auch [10]. Außerdem fand man heraus, dass Kunden, sobald es sich um sensiblere Produkte oder Suchanfragen handelt, ihre Privatsphäre ausgeprägter schützen wollen, als bei Alltäglichem. Wie sich die Person dabei genau verhalten hat, war je nach Land und Person zwar unterschiedlich, es konnten jedoch Tendenzen festgestellt werden.

Aus diesem Vorwissen wurden in [1] diese fünf Hypothesen aufgestellt, die es zu bestätigen oder zu widerlegen galt:

H1 – The price of privacy-enhancing features and the proportion of users enabling them are negatively associated.

H2 – The more sensitive the search task, the more likely users will enable privacy-enhancing features.

H3 – The more sensitive the search task, the less likely users will enable privacy-invasive features.

H4 – Users who are more concerned about privacy will enable privacy-enhancing features more often.

H5 – Users who consider privacy-enhancing features more important will enable them more often.

3.2 Ergebnisse des Experiments

Bevor die Ergebnisse des Experiments dargestellt werden, soll hiermit noch angemerkt werden, dass bei der Durchführung verschiedene Punkte aufgefallen sind, die in zukünftigen Feldversuchen verändert werden sollten. So sollte man zum Beispiel beachten, dass Probanden im Verlauf eines Experiments ihr Verhalten ändern können und unterschiedliche Arten für Einstellungen zu Bezahlen auch ein unterschiedliches Verhalten gegenüber der Auswahl von Optionen mit sich bringt (z.B. ein Abonnement). Siehe hierzu Seite 12 unter [1].

3.2.1 Bestätigte Hypothesen

Nachdem das Experiment durchgeführt und ausgewertet war, ist ein Ergebnis stark aufgefallen. Viele der Einstellungsmöglichkeiten, die eine positive Auswirkung auf die Privatsphäre ausübten, wurden, solang sie kostenlos waren, von sehr vielen Probanden genutzt, jedoch verringerte sich die Anzahl der Nutzer drastisch, sobald man dafür bezahlen musste. So fiel zum Beispiel die Anzahl der Personen, die die „Verlauf nicht speichern“-Funktion nutzten, auf ein Viertel, sobald 2 Credits dafür verlangt wurden. Somit zeigte sich, dass die Einführung von Kosten die Nutzer herausfiltert, die Wert auf diese Funktionen legen. Zum Beispiel fiel die Nutzung der „Kein Verlauf“-Funktion von 58% auf 15% und die Nutzung der „Keine Weitergabe an Dritte“-Funktion von 79% auf 16%, als diese mit Kosten verbunden waren. Durch dieses Ergebnis konnte Hypothese 1 bestätigt werden.

Weiterhin konnte Hypothese 2 und Hypothese 3 ebenfalls voll bestätigt werden. So bemerkte man einen deutlichen Anstieg der Nutzung von den Funktionen „Verlauf nicht Speichern“ und „Keine Weitergabe an Dritte“, sobald es um sensible Suchanfragen ging. Bei der Veröffentlichung von Suchanfragen auf Twitter, war das Ergebnis genau umgedreht. Umso sensibler die Suchanfrage war, desto seltener wurde die „Auf Twitter veröffentlichen“-Funktion genutzt. Während des Experiments war auch aufgefallen, dass sobald die Funktionen nicht mehr kostenlos verfügbar waren, diese auch bei sensibleren Anfragen nicht mehr aktiviert wurden.

Hypothese 5 konnte hingegen nur teilweise bestätigt werden. Die Nutzer für die eine „Verlauf nicht Speichern“-Funktion laut ihrer Angabe wichtig sei, haben diese nicht häufiger oder weniger häufig eingesetzt, als andere Teilnehmer. Eine Bestätigung der Hypothese konnte man aber bei der „Nicht an Dritte weitergeben“-Funktion registrieren. Hier war der Anteil der Personen, die für die Wichtigkeit gestimmt haben, höher als bei der anderen Gruppe, die nicht so viel Bedeutung in diese Funktion legen. Dennoch wurde ersichtlich, dass Nutzer ungern Geld für ihre Sicherheitseinstellungen ausgeben, auch wenn sie vorher angaben, diese Funktionen seien ihnen wichtig. [1]

3.2.2 Widerlegte Hypothesen

Neben der teilweise widerlegten beziehungsweise bestätigten Hypothese 5 (dessen Ergebnis in Kapitel 3.2.1 beschrieben wird), konnte Hypothese 4 vollständig widerlegt werden. So haben die Nutzer zwar angegeben, dass großer Wert auf Datenschutz gelegt wird, jedoch wurden, unabhängig von den Creditkosten, die datenschützenden Funktionen, wie zum Beispiel „Verlauf nicht speichern“ und „Keine Weitergabe an

doi: 10.2313/NET-2014-08-1_17

Dritte“, nicht häufiger genutzt. Bei weiterem Nachfragen fiel aber auf, dass die Probanden keine genaue Auskunft darüber geben konnten, welche Optionen zum Datenschutz beitragen und welche nicht. So wurde das Deaktivieren von Werbung auf der Ergebnisseite der Suchanfrage schon von vielen Nutzern als eine Erhöhung des Datenschutzes gesehen. Außerdem ist bei der Auswertung aufgefallen, dass mehrere Nutzer zwar angaben, ihnen sei Datenschutz wichtig, jedoch wurden ihre Aussagen von ihren Handlungen und Einstellungen widerlegt. [1]

3.3 Umfrage

Wie auch bei Kapitel 3.2 wird hiermit verdeutlicht, dass die hier genannte Umfrage aufgrund der geringen Anzahl an Teilnehmern, selbst nicht repräsentativ, jedoch soll damit ein Vergleich zum Experiment zur Verfügung gestellt und ein Bezug zwischen Aussagen und Handeln verschiedener Personen dargestellt werden.

3.3.1 Aufbau

Um die Hypothesen und Ergebnisse aus dem im Absatz 3.1. und 3.2 vorgestellten Experiment vergleichen zu können, wurde eine, an das Experiment angepasste, Internetumfrage erstellt, die sich hauptsächlich an den Thesen und Einstellungsmöglichkeiten des Experiments orientiert. Diese Umfrage wurde an 19 Personen, mit unterschiedlichem Alter und Lebenslage verteilt, welche diese vollständig ausgefüllt und zurückgesendet haben. In der Umfrage wurden verschiedene Fragen zu den verwendeten Sucheinstellungen gestellt. Die Fragestellung wurde teilweise so angepasst, dass sich eine Vergleichbarkeit zum Experiment ergab. Dennoch konnte man eigene Kommentare zu den jeweiligen Fragestellungen abgeben, welche auch oft genutzt wurden und so das Gesamtbild der Ergebnisse verbesserten. Am Anfang der Umfrage wurde nach der verwendeten Suchmaschine, der Nutzung der „Verlauf speichern“-Funktion und den gewünschten Einstellungsmöglichkeiten gefragt. Man konnte jeweils angeben, ob die in Tabelle 3 angegebenen Möglichkeiten genutzt werden oder nicht. Diese Frage wurde in zwei Teile aufgespalten. Im ersten Teil wurde angenommen, dass die Nutzung der Einstellungsmöglichkeiten kostenfrei zur Verfügung gestellt wird. Im zweiten Teil hingegen, sollte man für die Aktivierung der Einstellungsmöglichkeiten bezahlen. Ein Beispiel für eine Fragestellung ist: „Würden Sie eine No-Ads-Funktion verwenden?“, diese hatte die Antwortmöglichkeit „Ja, ich würde No-Ads verwenden.“, beziehungsweise „Nein, oder es ist mir egal.“. Der nächste Teil der Umfrage beschäftigte sich mit der Thematik der Seitenfinanzierung, welche Optionen die Befragten akzeptieren würden. Die möglichen Themen waren: „Werbung auf der Ergebnisseite“, „Sponsored Links“, „Die Suche würde auf Twitter veröffentlicht“ und „Die Suchergebnisse, auf die Sie klicken, werden gespeichert.“ Zum Schluss wurden noch nach der persönlichen Einstellung gegenüber Datenschutz und der Häufigkeit der Verwendung von Anonymisierungsdiensten und VPN-Verbindungen gefragt. [3]

3.3.2 Ergebnis und Vergleich mit dem Experiment

Sowohl im Experiment, als auch in der Umfrage stellte sich heraus, dass den Nutzern, laut eigener Angabe, Datenschutz sehr wichtig ist, aber das Handeln diese Aussage nicht unterstützt. So werden persönliche Daten eher preisgegeben, sobald der Schutz entweder etwas kostet oder mit Aufwand verbunden ist.

In Tabelle 2 und 3 werden jeweils die Prozentzahlen der Nutzer einer bestimmten Einstellung gegeben. Dabei steht ein + für die Aktivierung beziehungsweise Nutzung des Features und ein – für die Deaktivierung beziehungsweise nicht Nutzung eines Features, jeweils im Vergleich, ob es kostenlos oder kostenpflichtig angeboten wurde. Ein großer Unterschied zwischen dem Experiment und der Umfrage ist beim Thema „Weitergabe an Dritte“. Dabei ist der Prozentsatz im Experiment von 79% auf 16% gefallen, sobald dafür Geld verlangt wurde. In der Umfrage jedoch sank der Prozentsatz von 100% nur auf 74%. Somit würden die Nutzer auch in einer gewissen Form dafür bezahlen, dass ihre Daten unter Verschluss bleiben. Auch fällt hierbei auf, dass die Nutzer ihre Daten nur an Personen oder Firmen weitergeben wollen, denen sie persönlich vertrauen. In den Ergebnissen lässt sich auch erkennen, dass das Entfernen von Werbung und Sponsored-Links, beziehungsweise das verbessern des Suchergebnisses und das Hervorheben von Suchanfragen im Ergebnis, zwar ein gern gesehenes Feature ist, jedoch vernachlässigt wird, sobald dafür Kosten entstanden wären. Beim Thema „Miteinbeziehen des Standorts in der Suchanfrage“ fällt auf, dass dieses Feature im Experiment nach den Zahlen in der Tabelle von fast allen genutzt wurde und bei der Umfrage jedoch kaum. Dies lässt sich aber leicht durch das Entstehen der Zahlen erklären. So wurde im Experiment nur berücksichtigt, ob diese Funktion min. einmal oder häufiger genutzt wurde. Somit, wurde jede Person gezählt, die auch nur ein einziges Mal, diese Funktion verwendet hat. In der Umfrage ist dies umgedreht. Hierbei wurde die Ortung bei der Ergebnissuche im Allgemeinen abgelehnt, jedoch wenn zum Beispiel Restaurants in der Nähe gesucht werden sollten, würden die meisten Befragten diese Funktion aktivieren, sofern sie kostenlos zur Verfügung gestellt wird.

Des Weiteren kann man an den Ergebnissen sehen, dass die Nutzer ihre Sucheinstellungen nach ihrer Suchanfrage anpassen. Das erkennt man sowohl an den Einstellungen bei empfindlichen Suchanfragen im Experiment, als auch in den Feedbacktexten der Umfrage. Ein Beispiel dafür ist die Safe-Search-Funktion. Diese wurde im Experiment etwa gleich oft genutzt, egal ob dafür Geld verlangt wurde oder nicht. Laut Umfrage wurde diese Funktion, solang Sie kostenlos war, von 21% immer und von 53%, wenn Kinder und Jugendliche geschützt werden sollten, genutzt. Sobald Geld dafür verlangt wurde verschob sich dieser Bereich zu Nein, jedoch war das Thema Jugendschutz immer noch wichtig, wie man im Feedback lesen konnte.

Zum Thema Seitenfinanzierung, verhielten sich Experiment und Umfrage ähnlich, bis auf eine Einstellungsmöglichkeit. Im Experiment stimmten 75% einer Veröffentlichung ihrer Suche auf Twitter zu. Nach der Umfrage wurde die Veröffentlichung strikt abgelehnt, siehe Tabelle 4. Die restlichen 16% würden nur zustimmen, sofern die Veröffentlichung anonym oder mit großen Vorteilen verbunden wäre. [1][3]

doi: 10.2313/NET-2014-08-1_17

Tabelle 2. Nutzung von Einstellungen im Experiment (in %)

	Kostenlos		Mit Credits	
Einstellung	+	-	+	-
Speichere meinen Suchverlauf nicht.	58	42	15	85
Entferne Werbung auf der Ergebnisseite.	94	6	19	81
Gib meine Suchanfragen nicht an Dritte weiter.	79	21	16	84
Verbessere das Suchergebnis.	~61	~39	~52	~48
Hebe die Suchanfrage in den Ergebnissen hervor.	~50	~50	~45	~55
Kopple die Suchergebnisse an meinen Standort.	~90	~10	~80	~20
Verwende die sichere Suche Funktion.	~69	~31	~69	~31
Veröffentliche meine Suche auf Twitter.	~75	~25	~75	~25
Speichere, auf welche Ergebnisse ich klicke.	~88	~12	~80	~20

Tabelle 3. Nutzung von Einstellungen nach der Umfrage (in %)

	Kostenlos		Kostenplichtig	
Einstellung	+	-	+	-
Entferne Werbung auf der Ergebnisseite.	89	11	42	58
Sponsored-Links verbieten	47	53	26	74
Gib meine Suchanfragen nicht an Dritte weiter.	100	0	74	26
Verbessere das Suchergebnis.	74	26	32	68
Hebe die Suchanfrage in den Ergebnissen hervor.	74	26	21	79
Kopple die Suchergebnisse an meinen Standort.	11	89	5	95
Verwende die sichere Suche Funktion.	74	26	49	53

Tabelle 4. Möglichkeiten der Seitenfinanzierung in der Umfrage (in %)

Finanzierungsmöglichkeit	+	-
Werbung auf der Ergebnisseite	84	16
Sponsored Links auf der Ergebnisseite	74	26
Suchergebnisse auf Twitter veröffentlichen	16	84
Suchergebnisse, auf die Sie klicken werden gespeichert	64	37

Die Umfrage ging noch auf zwei weitere Themen ein, die so nicht im Experiment behandelt wurden. Zum einen, welche Suchmaschinen verwendet werden, zum Anderen wie wichtig Datenschutz ist und ob Schutzmaßnahmen ergriffen werden.
Dabei stellte sich, wie in Abbildung 1 gegeben, heraus, dass der Hauptanteil eindeutig bei Google liegt. Obwohl es in der Datenschutzerklärung von Google genau beschrieben ist, dass Google alle Suchanfragen detailreich abspeichert [11], verwenden nur 3 Personen Startpage beziehungsweise nur 2 DuckDuckGo, welche die Google Search Engine zur Suchergebnisverbesserung verwenden, aber keine Daten-speicherung vornehmen. [3]

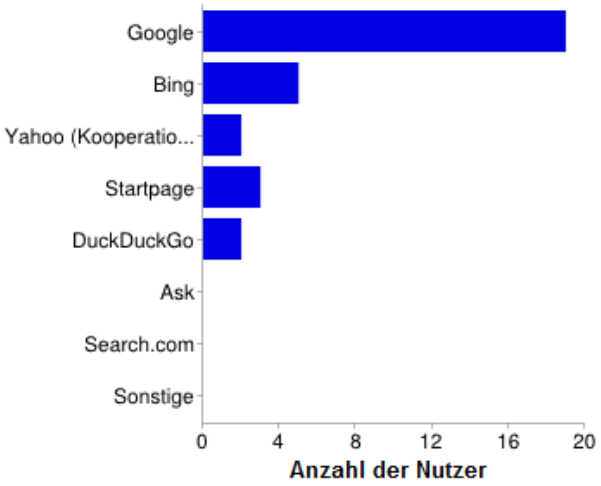


Abbildung 1: Auswertung zur Frage: „Welche Such-maschinen verwenden Sie.

Außerdem wurde nach der Bedeutung von Datenschutz gefragt und der Häufigkeit der Verwendung von Anonymisierungs-software, gegeben in Abbildung 2 und 3. Beim Vergleich der beiden Diagramme fällt auf, dass diese Spiegelungen voneinander sind. So ist den Befragten zwar ihr Datenschutz sehr wichtig, aber sie nutzen meistens keine Software zum Verschleiern ihrer Identität. Genauso sieht es bei der Verwendung von VPN-Verbindungen aus. So verwenden 64% entweder gar keine gesicherte Verbindung oder nur, wenn sie müssen, zum Beispiel das Einwählen in das Uni- oder Firmennetz. Nur 37% nutzen eine gesicherte Verbindung, sobald sie mit einem öffentlichen, für alle zugänglichen Netz verbunden

doi: 10.2313/NET-2014-08-1_17

sind. Die kaum vorhandene Sicherheit dieser öffentlichen Verbindungen wurde in den Medien bereits ausgiebig diskutiert. Als Beispiel siehe hierzu [12]. Dennoch wird die Möglichkeit einer VPN-Verbindung nicht häufig genutzt, da dies entweder mit Kosten oder mit Aufwand verbunden ist. [3]

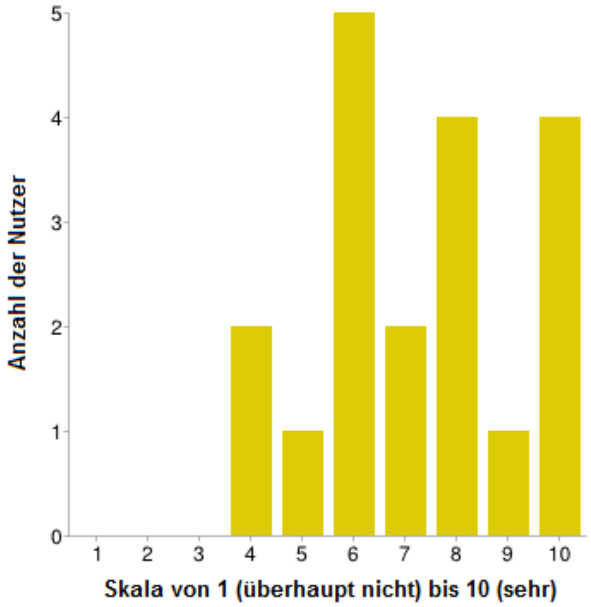


Abbildung 2: Auswertung zur Frage: „Wie wichtig ist Ihnen Datenschutz?“

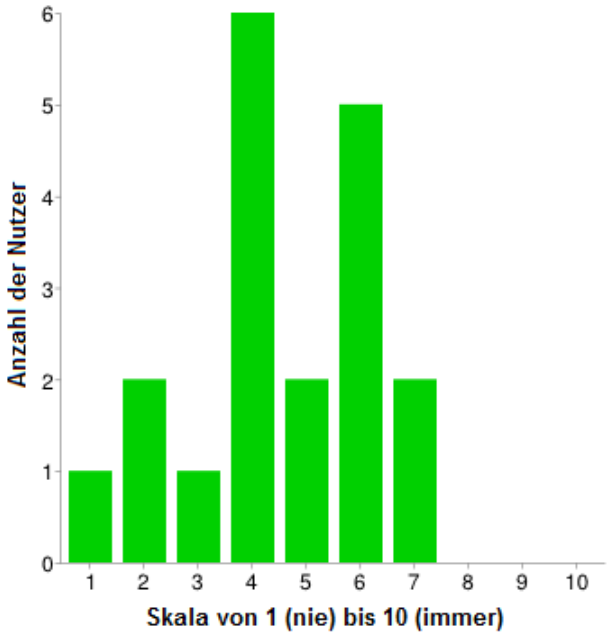


Abbildung 3: Auswertung zur Frage: „Wie oft verwenden Sie Anonymisierungssoftware?“

4. ZUSAMMENFASSUNG

Der Begriff Datenschutz beziehungsweise Schutz der persönlichen Daten im Internet ist in den Medien bereits seit längerer Zeit ein fester Bestandteil. Man müsste somit annehmen, dass ein Mindestmaß an Anonymisierung und somit dem Verschleiern von eigenen persönlichen Daten standardmäßig bei der Nutzung des Internets und hier vor allem auch bei der Suche im World Wide Web vorhanden sein müsste.

Sowohl nach dem Experiment, als auch in der Umfrage, stellte sich heraus, dass den Nutzern von Suchmaschinen, laut eigener Aussagen, sehr viel an Datenschutz liegt. Aber sobald Kosten oder ein gewisser Aufwand entstehen, tritt der Schutz persönlicher Daten oft in den Hintergrund. Außerdem ist aufgefallen, dass anonymisierte Suchmaschinen eher unbekannt sind und die bekannten Firmen, wie Google und Microsoft (Bing), bisher keine Möglichkeit bieten für die Anonymisierung eigener Daten zu bezahlen. Somit gibt es für den normalen Internetnutzer nur die Möglichkeit, durch die Preisgabe seiner Daten, ein akkurates Suchergebnis zu erhalten.

doi: 10.2313/NET-2014-08-1_17

5. REFERENZEN

[1] Sören, Preibusch. 2013. The value of privacy in Web search. Microsoft Research Cambridge UK. The twelfth workshop on the economics of information security in WEIS 2013. http://weis2013.econinfosec.org/papers/PreibuschWEIS2013.pdf. Zuletzt aufgerufen: 30.07.2014

[2] K. Keenoy und M. Levene. Personalisation of Web search. In B. Mobasher und S. S. Anand, Editoren, Intelligent Techniques for Web Personalization, Volume 3169 of Lecture Notes in Computer Science, Seiten 201-228. Springer Berlin Heidelberg, 2005. http://link.springer.com/chapter/10.1007%2F11577935_11

[3] Eigene durchgeführte Umfrage Mai 2014

[4] Urteil des Europäischen Gerichtshofs in der Rechtssache C-131/12 vom 13. Mai 2014. http://curia.europa.eu/juris/document/document.jsf?text=&docid=152065&pageIndex=0&doclang=DE&mode=req&dir=&occ=first&part=1&cid=258738, Zuletzt aufgerufen: 10.06.2014

[5] Google Formular zur Löschung von Suchanfragen. 2014. https://support.google.com/legal/contact/lr_eudpa?product=websearch&hl=de. Zuletzt aufgerufen: 30.07.2014

[6] Tor-Project. Juli 2014. https://www.torproject.org/about/overview.html.en. Zuletzt aufgerufen: 30.07.2014

[7] JonDoNym. Juli 2014. http://www.anonym-surfen.de/prinzip.html. Zuletzt aufgerufen: 30.07.2014

[8] Janice Tsai, Serge Egelman, Lorrie Cranor, and Alessandro Acquisti, "The Effect of Online Privacy Information on Purchasing Behavior: An Experimental Study," in WEIS 2007.

[9] Alastair R. Beresford, Dorothea Kübler, and Sören Preibusch, "Unwillingness to pay for privacy: A field experiment," Economics Letters, vol. 117, no. 1, Seiten 25-27, 2012.

[10] Nicola Jentzsch, Sören Preibusch, and Andreas Harasser, "Study on monetising privacy. An economic model for pricing personal information," European Network and information Security Agency (ENISA), 2012.

[11] Datenschutzerklärung Google. Stand 31. März 2014. Absatz: Von uns erhobene Informationen. https://www.google.de/intl/de/policies/privacy/. Zuletzt aufgerufen: 30.07.2014

[12] Artikel: Die Hotspot Falle. Heft C't 1/2012. http://heise.de/-1394646. Zuletzt aufgerufen: 30.07.2014

[13] Bundesdatenschutzgesetz §3. Stand 2014. http://www.gesetze-im-internet.de/bdsg_1990/__3.html. Zuletzt aufgerufen: 31.07.2014

[14] Artikel: Gelöscht und Gut? von Alexander Drößler. http://www.tagesschau.de/wirtschaft/google-urteil-100.html. Zuletzt aufgerufen: 31.07.2014

[15] TUM Vorlesung „IT-Security". Prof. Dr. Eckert, Dipl.-Inf. Thomas Kittel. Wintersemeseter 13/14

[16] TUM Vorlesung „Network Security". Prof. Dr.Ing. Georg Carle, Dr. Heiko Niedermayer, Ralph Holz. Wintersemester 13/14

[17] Google Trends. Juni 2014. https://www.google.de/trends/. Zuletzt auftgerufen: 01.08.2014

DNS Privacy

Patrick Werneck
Betreuer: Johannes Naab
Seminar Innovative Internettechnologien und Mobilkommunikation, SS2014
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
werneck@in.tum.de

KURZFASSUNG

Im Domain Name System (DNS) existieren sowohl auf Seiten der Clients als auch auf Seiten der Betreiber erhebliche Sicherheitslücken im Bezug auf die Vertraulichkeit und Privatheit der jeweils eigenen Daten. Der Informationsfluss von Clients, die einen Domain Name auflösen wollen, findet unverschlüsselt statt und wird meist durch mehrere Server geleitet. Serverbetreiber und Angreifer, die die Nachrichten zwischen Client und primären DNS-Server mitschneiden können, haben so Zugriff auf Informationen zu den verwendeten Domain Names, sowie deren Häufigkeit. Auf Seiten der DNS Serverbetreiber öffnet die Verwendung der standardisierten DNS Sicherheitserweiterung DNSSec Angreifern die Möglichkeit die komplette Zone des Betreibers auszulesen. Mit Verschlüsselung der DNS Nachrichten können nur die auf dem Übertragungsweg auftretenden Probleme behoben werden. Weitere Lösungsansätze, die einen Umbau des Domain Name Systems erfordern, greifen auch diese Probleme auf, sind jedoch aufgrund ihrer schlechten Skalierbarkeit nicht realistisch einsetzbar.

Schlüsselworte

dns, privacy, DNSCurve, DNSCrypt, cPIR, PPDNS, Range-Query, QNAME Minimisation

1. EINLEITUNG

Das Domain Name System (DNS) ist eine hierarchische verteilte Datenbank die verschiedene Schlüssel-Wert-Paare speichern und zur Verfügung stellt. Unter anderem werden menschlich lesbare und einprägsame Namen in IP-Adressen umgewandelt, die für das Versenden und Routen von Paketen durch das Internet benötigt werden. Dadurch ist das DNS zu einem integralen Bestandteil des Internets geworden. Nach Betrachtung des Protokollverlaufs, der in Abschnitt 2 beschrieben wird, ist leicht zu erkennen, dass das ursprüngliche DNS keinerlei Sicherheitsmechanismen besitzt. Dadurch kann jede DNS-Nachricht mitgelesen und bei aktiven Angreifern auch beliebig verändert werden. Während die Probleme der Datenintegrität und Authentizität bereits durch die Erweiterung Domain Name System Security Extensions (DNSSEC) behandelt werden, wird der Aspekt der Privatsphäre und Vertraulichkeit der Nachrichten bisher nicht beachtet. Diese Probleme entstehen jedoch nicht nur durch die Möglichkeit des Angreifers Nachrichten mitzulesen, sondern auch durch die Nameserver selbst, da auch diese alle DNS-Nachrichten sehen können. So ist es möglich sowohl die angefragte Domain als auch die Häufigkeit dieser Anfragen zu bestimmen. Ersteres ermöglicht zusätzlich zur reinen Information eine Zensur der erreichbaren Domainnamen durch den Betreiber des Nameservers für seine Nutzerbasis. Letzteres kann für die Überwachung des Netzes auf neue sich schnell verbreitende Seiten verwendet werden. Diese können dann analysiert und dem Ergebnis nach blockiert werden. Es kann außerdem eine Statistik für einzelne Nutzer angelegt werden, durch die Rückschlüsse auf die Lebensweise gezogen werden kann.

In diesem Artikel werden verschiedene Lösungsansätze für die Datenschutzprobleme aufgezeigt. Dafür wird zuerst in Abschnitt 2 das DNS Protokoll analysiert, woraufhin die einzelnen Angriffsmöglichkeiten auf die Privatsphäre aus Sicht der Endnutzer und Zonenbetreiber in Abschnitt 3 vorgestellt werden. Abschnitt 4 führt darauf aufbauend einzelne Lösungsansätze für einzelne Teilprobleme auf, die in Abschnitt 5 verglichen werden und ein Ausblick für vielversprechende Kandidaten gegeben wird.

2. DNS ÜBERBLICK

Der Namensraum des Domain Name System ist in einer Baumstruktur gegliedert. Jeder Knoten in diesem Baum stellt eine Domain dar und besitzt bis auf die Wurzel ein nichtleeres Label. Der zu jedem Knoten zugehörige Fully Qualified Domain Name (FQDN) bildet sich durch Aneinanderreihung aller Label der Knoten vom Blatt bis zur Wurzel mit einem Punkt („.“) als Separatorzeichen. Knoten auf der ersten Ebene unter dem Wurzelknoten werden als Top-Level Domains (TLD) bezeichnet, alle folgenden Ebenen entsprechend ihrer Ebenentiefe (Second-Level Domain, Third-Level Domain, usw.). Der in Abbildung 1 benutzte Domainname *www.example.com* ist also der Knoten *www.example.com* mit dem Label *www* und dem Elternknoten *example.com* (Second-Level Domain), welcher *com* als Elternknoten hat (TLD). Der Knoten *com* stammt wiederum von *root*, der Wurzel des Domain Name Systems, ab.

Innerhalb des Domain Name System Baums bilden einzelne Teilbäume eine Zone. Für jede Zone ist dabei mindestens ein autoritativer Nameserver verantwortlich um eingehende DNS Anfragen zu beantworten und die Zone zu verwalten. Die Informationen zu den einzelnen Knoten der Zone werden dabei in Resource Records (RR) abgespeichert. Jedes RR besteht aus einen Typ, den dazugehörig entsprechend kodierten Daten, ein Namensattribut, welches den zugehörigen Domainnamen des RR bestimmt und eine TTL, die die Gültigkeit des RR bestimmt, enthalten.Mit dem RR Typ A wird ein Domainnamen auf eine IPv4-Adresse abgebil-

doi: 10.2313/NET-2014-08-1_18

det. Alle zu einer gegebenen Anfrage passenden Resource Records werden zusammengefasst RRset genannt. Um ein gewisses Maß an Ausfallsicherheit zu erreichen können mehrere autoritative Nameserver betrieben werden. Die Zoneninformationen werden dabei von einem Master-Server auf Slave-Server mit Hilfe der Zonentransferprotokollen IXFR und AXFR, die hier nicht weiter behandelt werden, verteilt.

Neben den autoritativen Nameserver gibt es noch die rekursiven Nameserver. Im Gegensatz zu den autoritativen Nameservern verwalten diese keine eigene Zone, sondern bieten Clients, die DNS Anfragen einleiten (Stub Resolver), eine rekursive Namensauflösung an. Das heißt der rekursive Nameserver vollzieht vom Wurzelknoten, bzw. den Root-Servern an bis zum vom Client angefragten Blattknoten alle nötigen Anfragen. Um die Last der autoritativen Nameserver zu reduzieren und Zeit zu sparen, werden die erhaltenen Antworten zwischengespeichert, und erst mit Ablauf der in den RR enthaltenen TTL wieder verworfen. Dadurch muss nicht für jede Anfrage erneut der gesamte Baum durchlaufen werden, und der Client erhält ohne erhöhte Latenz die angefragte Antwort. Eine Anfrage besteht dabei jeweils aus einem Domainnamen (QNAME) und einem Anfragetype des angefragten Domainnamens. Der Anfragetype ist bis auf wenige Ausnahmen der Typ des RR welches als Antwort zurückgegeben werden soll.

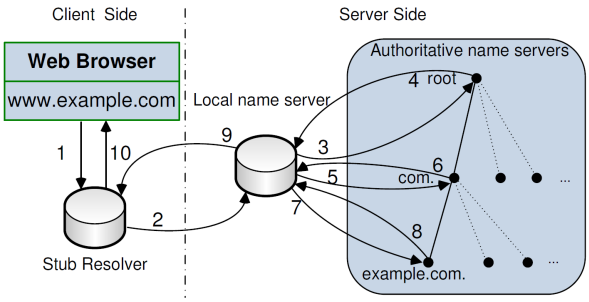


Abbildung 1: Rekursive Namensauflösung der Domain *www.example.com* [12]

Zur Veranschaulichung des rekursiven Namensauflösungsprozesses werden im Folgenden alle benötigten Einzelschritte von der Clientanfrage ausgehend anhand von Abbildung 1 beschrieben. (1) Eine Applikation - in diesem Fall ein Browser - will eine Verbindung zu *www.example.com* aufbauen, und fragt dafür die IP beim Stub Resolver, dem lokalen Dienst des Betriebssystems, ab. Dieser schickt, da Stub Resolver im Allgemeinen über keinen Cache verfügen, in (2) eine DNS Anfrage mit der Frage „Wie lautet die IP von *www.example.com*“ an den konfigurierten rekursiven Nameserver. Sofern dieser die Antwort noch nicht im Cache liegen hat und auch keine anderen Nameserver kennt, die für diese Domain zuständig sind, leitet der rekursive Resolver in (3) die Anfrage des Clients an einen Root-Nameserver weiter. Dieser Antwortet in (4) mit den Domain-Namen und IPs der zuständigen autoritativen Nameserver der *com* Zone. (5) Der rekursive Resolver leitet nun die Anfrage des Clients an den autoritativen Nameserver der *com* Zone weiter, welcher in (6) den rekursiven Nameserver an den autoritativen Server der *example.com* Zone delegiert. Nach der erneuten Weiterleitung der Clientanfrage in (7) antwortet der autoritative Nameserver für die *example.com* Zone in (8) mit der IP Adresse des Hosts *www.example.com*. Da der rekursive Resolver nun die Adresse kennt, schickt er in (9) die Antwort „*www.example.com* hat die Adresse *192.0.2.42*“ an den Stub-Resolver, welcher wiederum in (10) der Applikation die IP-Adresse liefern kann und sich diese mit dem Host *www.example.com* verbindet.

Da das gesamte DNS Protokoll nur Klartextnachrichten versendet, bietet es zunächst keine Mechanismen zur Sicherung von Integrität, Authentizität und Vertraulichkeit. Die Erweiterung DNSSEC setzt dabei die ersten beiden Ziele um, verzichtet aber ausdrücklich auf den Aspekt der Vertraulichkeit, da diese nur öffentlich zugängliche Informationen enthält. DNSSEC soll die gute Skalierbarkeit des DNS Systems beibehalten, indem alle nötigen kryptographischen Operationen vorberechnet werden. So wird mit Hilfe von Public Key Kryptographie für jedes ursprüngliche RRSet in der Zonendatei ein zusätzliches RR vorberechnet, welches die Signatur des originalen RRSet enthält. Es werden außerdem die zusätzlichen RR NSEC beziehungsweise NSEC3 eingeführt, um die nicht Existenz eines Domain Namens beweisen zu können. Bei NSEC werden lexikographisch geordnet Paare von Domain Namen gebildet, so dass ein Ring gebildet wird. Sollte ein Name angefragt werden, der nicht existiert, so wird mit dem NSEC RR geantwortet, welches das Paar enthält zwischen den der angefragte Name liegt, und damit die nicht Existenz des Domain Namens sichergestellt ohne zur Laufzeit Antworten signieren zu müssen.

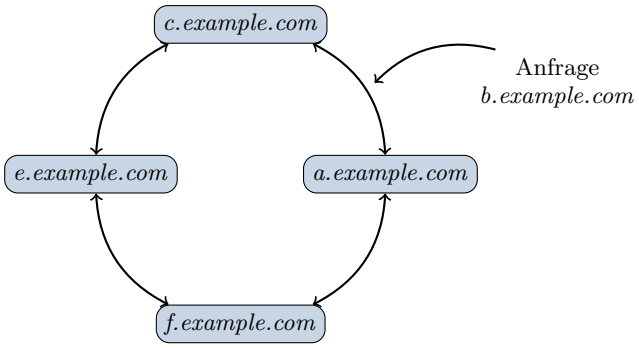


Abbildung 2: Anfrage einer nicht existenten Domain auf eine NSEC Ringstruktur mit vier Domainnamen

In Abbildung 2 ist eine solche NSEC Struktur mit vier Domainnamen gebildet. Bei der Anfrage der nicht existenten Domain *b.example.com* wird das signierte NSEC RR mit dem Paar *a.example.com* und *c.example.com* als Antwort gesendet und damit die Nichtexistenz des angefragten Namens sichergestellt. Bei NSEC3 werden anstatt der Klartext Hostnamen gehashte Domainnamen zur Bildung von sortierten Paaren verwendet, um den in Abschnitt 3.2 beschriebenen Angriff zu erschweren.

3. ANGRIFFSANALYSE

Aufbauend auf der vorhergehenden Analyse des DNS Protokolls und der Erweiterung DNSSec werden im folgenden die verschiedenen Angriffsmöglichkeiten auf die Privatheit der DNS-Daten sowohl der Sicht des Endnutzer als auch aus der Sicht des Zonenbetreibers analysiert.

doi: 10.2313/NET-2014-08-1_18

3.1 Endnutzer

Wird eine DNS-Anfrage an einen Nameserver verschickt, ist dies mit dem Ziel verbunden, sich mit dem entsprechenden Dienst zu verbinden. Aus Sicht der Endnutzer ist nicht der Inhalt der Antwort an sich die schützenswerte Information, sondern die Tatsache, das dieser einen bestimmten Dienst nutzen will. So ist beispielsweise nicht die öffentlich verfügbare Information, dass die Seite *foo.example.com* die die IP 192.0.2.42 hat schützenswert, sondern das der Nutzer diese Seite besuchen will. Auch kann eine DNS Anfrage Informationen über die verwendete Software geben, indem nach einem softwarespezifischen Serverrecord einer Domain wie *_bittorrent-tracker.* gesucht wird.

3.1.1 Auf dem Weg zu Ziel

Wie aus dem bereits in Kapitel 2 erläutert wurde, werden alle Nachrichten unverschlüsselt ausgetauscht. Dadurch können sie sowohl auf dem Weg vom Stub Resolver zum konfigurierten rekursiven Nameserver als auch von diesem zu den jeweiligen autoritativen Servern mitgeschnitten und gelesen werden. Da typischerweise Stub Resolver über keinen Cache verfügen, werden zwischen diesem und dem rekursiven Resolver jegliche DNS Anfragen von Anwendungen auf dem Client an den Server versendet. Ein Angreifer zwischen diesen beiden Stellen kann so ein komplettes Verlaufsbild des Clients, wie häufig und zu welcher Zeit verschiedene Dienste genutzt werden, erstellen. Im Gegensatz dazu sind die Möglichkeiten der DNS-Traffic-Analyse bei der Übertragung von rekursiven zu autoritativen Nameserver eingeschränkt, da das tatsächliche Anfragevolumen durch Caching im rekursiven Resolver und Anfragen anderer Clients nicht genau ermittelt werden kann. Zusätzlich verdeckt der Nameserver die Ursprungsadresse der Anfrage.

3.1.2 Durch Serverbetreiber

Zusätzlich zu der Möglichkeit Daten auf dem Weg zum jeweiligen Server mitzuschneiden können auch die Serverbetreiber selbst alle DNS Anfragen aufzeichnen und verwerten. Dabei kann der Betreiber eines rekursiven Nameserver analog zu Abschnitt 3.1.1 erneut den kompletten DNS Verlauf einzelner Nutzer auswerten. Auch Betreiber autoritativer Nameserver können erhalten den vollständigen Anfragenamen. Dadurch kann das durch Caching beschränkte Anfragevolumen bestimmt werden. Durch die statistische Auswertung der Lebensdauer der gegebenen Antworten und dem Anfrageintervall lässt sich jedoch das vollständige Anfragevolumen des rekursiven Nameservers, nicht jedoch des einzelnen Stub Resolvers, abschätzen [12].

Im Gegensatz zum Mitschneiden der DNS-Pakete auf dem Weg zu den jeweiligen Nameservern kann diese Schwachstelle nicht durch Schützen des Kommunikationskanals erreicht werden, da dieses Problem innerhalb der DNS Anfragebearbeitung auftritt.

3.2 Zonenbetreibers

Die Datenschutzaspekte der Zonenbetreiber beruhen darauf, dass der vollständige Zoneninhalt nicht öffentlich verfügbar sein soll, da dieser die komplette Netzwerkinfrastruktur auflistet. Angreifer, die den Inhalt der Zone kennen, können dadurch die Existenz wichtiger Ziele ausmachen. Ohne die Informationen aus der Zone kann es bedeutend schwieriger sein an diese Informationen zu gelangen, da so erst der korrekte Name des Hosts oder direkt die IP gefunden werden muss. So besteht zwar bei Einsatz von DNS ohne die Erweiterung DNSSEC zwar prinzipiell die Möglichkeit durch alle möglichen Namen zu iterieren, dies ist jedoch aufgrund der beschränkten Übertragungsrate nicht praktikabel. Außerdem erzeugt ein solcher Angriff eine große Datenflut, auf die der Zonenbetreiber entsprechend reagieren kann.

Durch die Einführung der Erweiterung DNSSEC wird, wie in Abschnitt 2 erläutert, mit dem neuen RR NSEC eine Klartext Ringstruktur aller vorhandenen Domainnamen gebildet. Wird nun ein Domainname abgefragt, der nicht existent ist, werden als Antwort zwei existente Domain Namen geliefert. Um die komplette Zone zu erhalten werden solange DNS-Anfragen mit Namen, die zwischen zwei existenten Domainnamen liegen, die noch nicht in der erhaltenen Ringstruktur verbunden sind, versendet, bis der komplette Ring empfangen wurde. Durch diese Methode werden nur so viele DNS-Anfragen benötigt wie in der Zone Einträge vorhanden sind [9].

Auch nachdem durch RFC5155 [11] das NSEC3 RR als Alternative für das NSEC RR eingeführt wurde, und die Domainnamen nicht mehr in Klartext, sondern in gehashter Form eine Ringstruktur bilden, kann auch hier ein analoger Angriff stattfinden. Da in diesem Fall jedoch statt der Klartextnamen nur die Hashwerte der existenten Domainnamen verfügbar sind, müssen diese noch lokal per Bruteforce erraten werden. Im Vergleich zu DNS ohne DNSSec wird so der limitierende Faktor des Bruteforceprozess von der Netzwerkgeschwindigkeit auf die des Prozessors verschoben, und dadurch erheblich beschleunigt. Zusätzlich werden nur noch wenige DNS-Anfragen benötigt und der Angriff dadurch für den Zonenbetreiber schwerer erkennbar gemacht [3].

4. LÖSUNGSMÖGLICHKEITEN

Nach der Analyse der verschiedenen Angriffsmöglichkeiten werden in diesem Abschnitt nun verschiedene Lösungsmöglichkeiten für einzelne Teilprobleme vorgestellt. Diese wurden dabei unabhängig voneinander entwickelt und können zum Teil miteinander kombiniert werden, um mehrere Teilprobleme aufzugreifen.

4.1 Verschlüsselung

In diesem Unterabschnitt werden verschiedene Lösungsansätze die Verschlüsselung der übertragenen Nachrichten oder Resource Records als Lösung vorschlagen. Verschlüsselungslösungen sind dabei für den Endnutzer nur in der Lage die in Abschnitt 3.1.1 beschriebenen Probleme durch Mitschnitt der Daten auf dem Übertragungsweg zu lösen, da die eigentlich DNS-Anfragen weiterhin durch den Nameserver verarbeitet werden müssen.

4.1.1 t-DNS: DNS über TCP mit TLS

Hu et al. [8] schlagen vor DNS-Nachrichten über TCP mit TLS verschlüsselt zu übertragen. Bereits in der Spezifikation von DNS in RFC1034 [14] und RFC1035 [13] wird eine TCP Implementierung vorgeschrieben. Aufbauend auf dieser Voraussetzung wird zum initialisieren der Verbindung eine normale DNS-Anfrage über TCP mit gesetztem TO-flag und einem beliebigen nicht aussagekräftigen Namen als Domain

doi: 10.2313/NET-2014-08-1_18

gesendet. Antwortet der Server mit einem TXT Record und RCODE=0, kann mit dem TLS-Sitzungsaufbau begonnen werden. Nachdem die TLS-Sitzung eingerichtet wurde, können DNS Anfragen in normalem Format gesendet werden.

Mit Einsatz von TCP und TLS werden pro Verbindungsaufbau zusätzliche vier Round Trip Times (RTT), Zeit, die eine Nachricht vom Client zum Server und wieder zurück, benötigt. Eine für den TCP Handshake, eine RTT für das Signalisieren der Unterstützung von TLS und weitere zwei RTT für den TLS-Sitzungsaufbau gebraucht. Um diesen erhöhten Aufwand und die Latenz zu minimieren, müssen aufgebaute Verbindungen für weitere DNS-Anfragen an den selben Server wiederverwendet werden. Um zusätzliche Wartezeit bei mehreren Anfragen zu verhindern müssen Clients diese direkt aufeinander folgend an den Server schicken und parallel vom Server bearbeitet werden. In Abbildung 3 werden

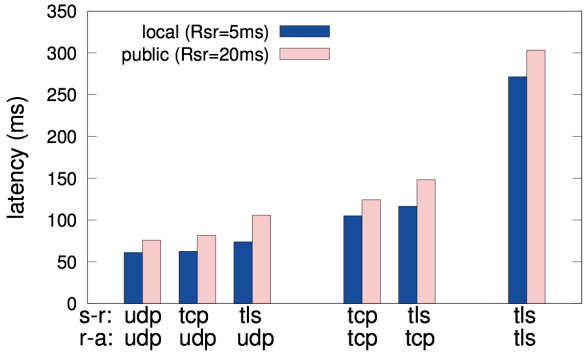


Abbildung 3: Durschnittliche Antwortzeiten von Stub Resolver zu rekursiven Nameserver des ISP und einem alternativen globalen mit verschiedenen Verbindungsarten zwischen Stub, rekursiv und autoritativen Resolvern [19]

verschiedene Verbindungsarten zwischen stub und rekursiven Resolver (s-r) und rekursiven Resolver und authoritativen Nameserver (r-a) verglichen. Dabei wird jeweils ein lokaler rekursiver Resolver des ISPs mit einer durchschnittlichen Antwortzeit (R_{sr}) von *5ms* und ein globaler mit einer durchschnittlichen Antwortzeit von *20ms* verwendet. Es ist gut ersichtlich, dass sich die zusätzlichen RTT die zur Erstellung einer TCP-TLS-Sitzung benötigt werden im Falle einer häufigen Wiederverwendung der Verbindung amortisieren. So steigt bei Verwendung von TCP-TLS anstatt von UDP als Verbindungsart von Stub zu rekursiven Resolver die Antwortzeit nur von durchschnittlich 62 ms auf 74 ms, da die Verbindung des Stub Resolvers für jede DNS Abfrage benutzt wird. Im Gegensatz dazu steigt bei zusätzlicher Verwendung von TCP-TLS für die Verbindung zwischen rekursiven und autoritativen Nameserver die Latenz von 74 ms auf 272 ms, da hier für viele verschiedene autoritative Nameserver eine Verbindung aufgebaut werden muss, die nicht mehr für weitere Abfragen benutzt wird. [8, 19]

Durch die Verwendung von TLS über TCP kann bei richtiger Wahl der TLS-Crypto-Suite Perfect Forward Secrecy hergestellt werden. Neben der Verschlüsselung der Nachricht wird bei TLS auch die Integrität dieser sichergestellt.

4.1.2 Confidential DNS

Confidential DNS ist ein von Wijngaards et al. [17] neu entwickeltes Kryptoprotokoll innerhalb des bestehenden DNS Protokolls. Dafür wird ein neues Resource Record vom Type ENCRYPT zur Keygenerierung und Übertragung von verschlüsselten Nachrichten eingeführt. Dabei werden mit einzelnen Flags nach dem ENCRYPT Record entsprechende Algorithmen, und die Art der folgenden Daten bestimmt. Das Protokoll kann sowohl für Kommunikation zwischen autoritativen und rekursiven Nameserver als auch zwischen rekursiven Resolvern und Stub Resolvern verwendet werden. Im folgenden Protokollverlauf wird die Partei, die die DNS-Anfrage startet mit Client bezeichnet, der Gegenüber mit Server.

Um eine verschlüsselte Übertragung zu starten, muss der Client zuerst Schlüsselmaterial vom Server anfordern, sofern kein passendes im Cache liegt. Dazu sendet er eine Anfrage mit dem Typ ENCRYPT und dem QNAME ".än den Server, welcher diese wiederum mit eigenem Schlüsselmaterial beantwortet. Nachdem der Client einen Schlüssel des Servers hat, sendet dieser erneut eine Anfrage mit dem Typ ENCRYPT und dem Rootlabel als Anfragedomain mit einem zusätzlichen ENCRYPT Record in der Additional Section, welches verschlüsselt die konkrete Anfrage und Schlüsselmaterial für die folgende Kommunikation enthält. Die Antwort des Servers enthält ähnlich zur Anfrage ein RR ENCRYPT, in dem die Antwort mit dem vorher gelieferten Schlüsselmaterial verschlüsselt ist. Sowohl der Server als auch der Client können erhaltene Schlüssel zwischenspeichern und wiederverwenden. Schlägt die Übertragung fehl oder der Server liefert einen Error, so muss der Client auf unverschlüsselte Übertragung umstellen.

Dieses Protokoll stellt weder Perfect Forward Secrecy her, noch prüft es die Integrität und Authentizität der Pakete. Dadurch muss DNSSEC innerhalb der verschlüsselten RR verwendet werden um Integrität sicherzustellen, wodurch das in Abschnitt 3.2 beschriebene Problem, der Möglichkeit die Zone komplett auszulesen, weiterhin besteht. Auch schützt Confidential DNS nur gegen passive Angreifer. Aktive Angreifer hingegen können leicht die Kommunikation auf unverschlüsseltes DNS zwingen, indem sie Nachrichten fehlerhaft manipulieren und dadurch auf normales DNS zurückgegriffen wird.

4.1.3 DNSCurve

DNSCurve ist ein von Daniel J. Bernstein [3] entwickeltes Protokoll für das Domain Name System, welches die Ziele Integrität und Vertraulichkeit für die Kommunikation zwischen autoritativen und rekursiven Nameservern umsetzt. Kryptographische Grundlage bildet die durch das Kryptoframework NaCl bereitgestellte elliptische Kurve. NaCl bietet außerdem Krytoboxen an, die aufbauend auf einem privaten, einem öffentlichen Schlüssel und einer Nonce, einem Wert der für jedes Schlüssel Paar nur ein einziges mal verwendet werden darf, Nachrichten authentifizierend ver- und entschlüsselt. Das gemeinsame Geheimnis wird über Diffie-Hellman auf elliptischen Kurven bestimmt [2].

Jeder DNS Server besitzt einen eigenen privaten und zugehörigen öffentlichen Schlüssel. Der öffentliche Schlüssel wird in ein neues Label in den Domainnamen des DNS Servers ko-

doi: 10.2313/NET-2014-08-1_18

diert und dadurch veröffentlicht. Wird eine Anfrage durch einen DNSCurve Client an einen DNSCurve kompatiblen DNS Server durchgeführt, so wird eine kryptographische Box mit dem eigenen privaten Schlüssel, dem öffentlichen Schlüssel des Empfängerservers und einer Nonce erstellt. In diese wird die originale Anfrage verpackt und zusammen mit dem eigenen öffentlichen Schlüssel und der Nonce and den Empfänger gesendet. Der Empfänger versucht nun mit dem empfangenen öffentlichen Schlüssel des Senders und der Nonce die Kryptobox zu entpacken. Sollte dabei ein Fehler auftreten, wird versucht, das Paket als normales DNS-Paket zu behandeln, um abwärtskompatibel zu bleiben. Konnte das Paket korrekt geöffnet werden, generiert der Server eine neue Nonce, verpackt die Antwort in einer Krytobox und versendet diese zusammen mit der ersten Nonce zurück an den Sender. Dieser versucht die Antwort zu öffnen, und verwirft diese, sofern die Kryptobox nicht korrekt entpackt werden kann. DNSCurve behält dadurch das normale DNS Protokoll bei und führt die Verschlüsselung ohne zusätzliche Nachrichten transparent ein. Durch das behandeln unverschlüsselter eingehender Pakete als normale DNS Anfragen, bleibt es abwärtskompatibel zum bestehenden DNS System, jedoch ohne verschlüsselte Verbindungen auf unverschlüsselte zu degradieren.

Da mit DNSCurve die komplette Verbindung auf Link-Level verschlüsselt ist, wirkt dies Methode effektiv gegen die in Abschnitt 3.1.1 beschriebenen Angriffsmöglichkeiten zwischen rekursiven und autoritativen Nameservern. Mit dem Einsatz von authentifizierender Verschlüsselung fällt die Notwendigkeit der Verwendung von DNSSEC weg und behebt dadurch die in Abschnitt 3.2 aufgezeigten Probleme im Bezug auf den Schutz des Zoneninhalts. Da die öffentlichen Schlüssel eines Nameserver in der jeweils darüberliegenden Zone eingetragen sind, kann auch die Authentizität der Schlüssel sichergestellt werden. Ein Problem besteht dabei lediglich die Schlüssel der Root-Zone zu authentifizieren. Außerdem bietet es, obwohl zur Erstellung des gemeinsamen Geheimnis ein Diffie-Hellman Verfahren verwendet wird, keine Perfect Forward Secrecy, da mit statischen Schlüsseln gearbeitet wird [2].

Neben der Spezifikation von DNSCurve, welche lediglich für den Einsatz zwischen rekursiven und autoritativen Nameservern ausgelegt ist, bietet OpenDNS mit DNSCrypt eine Erweiterung mit ähnlichen Designkriterien für die sichere Kommunikation zwischen Endnutzer und rekursivem Resolver an [15]. Auch hier bleibt jedoch die Angriffsmöglichkeit durch den Betreiber selbst bestehen, da dieser weiterhin alle Nachrichten sehen kann.

4.2 QNAME Minimierung

Stéphane Bortzmeyer [4] schlägt in einem IETF Draft vor die Anfragenamen auf das minimal Nötige zu reduzieren. Bei den aktuellen Implementierungen aller gängigen rekursiven Nameserver wird bei einer iterativen Namensauflösung bei jeder neuen Anfrage die komplette Ursprungsanfrage versendet. So wird an den Root-Server bei einer Anfrage der Domain *www.example.com* die vollständige Domain versendet, obwohl es ausreichend ist zu nach den Nameservern für die *com* Zone zu fragen. Durch die Minimierung der gefragten Domain (QNAME) in der Anfrage, sieht jeder Nameserver nurnoch die für ihn notwendigen und bekannten Details. Ein Problem bei dieser Methode ist die Bestimmung der Zonecuts, also jenen Stellen innerhalb der Domain, ab der ein anderer Nameserver zuständig ist, da dieser nicht an jedem Punkt sein muss. Dieses Verfahren ist mit der ursprünglichen Spezifikation des DNS-Protokolls in RFC1034 [14] und RFC1035 [13] kompatibel.

4.3 RangeQuery

Zhao et al. [18] schlagen vor den Anfragenamen zusammen mit einer Reihe anderer zufälliger Domainnamen zu versenden um die eigentliche Anfrage zu verschleiern. Dafür wird bei einer RangeQuery eine Liste $(name_1, ..., name_m)$ mit $m-1$ zufällig gewählten Domainnamen und dem geforderten Domainnamen versendet. Die $m-1$ zufälligen Domainnamen werden aus einer Menge mit n vorbestimmten gültigen Namen, die in unterschiedlichen vom Benutzer festgelegten Bereichen liegen, ausgewählt. So kann ein Angreifer nicht genau feststellen welcher Domainname das Ziel der Anfrage war. Während Zhao et al. [18] für jede DNS Anfrage die Liste der zufälligen Domain Namen neu zu bestimmen, um eine Erkennungswahrscheinlichkeit von $\frac{1}{m}$ für die echte Domain zu erreichen, widerlegen Lu und Tsudik [12] dies. Die Wahrscheinlichkeit sinkt mit der Anzahl an gesendeten RangeQuerys für die gleiche Domain, da bei mehreren Anfragen nur die Schnittmenge in allen Anfragen vorkommende Domainnamen in Frage kommen.

Abbildung 4 zeigt die Erkennungswahrscheinlichkeit nach k Anfragen mit jeweils einem Domain-Namen Pool der $n = 1024$ mit unterschiedlich großer Anfrageliste m. Es ist dabei leicht zu sehen, dass bei einer kleinen Anfrageliste bereits nach wenigen Wiederholungen die angefragte Domain mit hoher Wahrscheinlichkeit bestimmt werden kann. Lu und Tsudik [12] schlagen zur Lösung dieses Problems vor statt einer zufälligen Liste pro Anfrage eine Zufällige Liste pro Domain zu erstellen. Diese wird für alle weiteren Anfragen an diese Domain verwendet. Dies kann mit einem Pseudozufallszahlengenerator mit einem geheimen Startwert und der Domain erreicht werden.

RangeQuerys sind zusammen mit computational Private Information Retrieval (cPIR), welches im folgenden Abschnitt beschrieben wird, die einzige Methode um angefragte Domains vor rekursiven Nameservern zu verschleiern. Dabei muss bei RangeQuerys der erwünschte Sicherheitsgrad gegen den erzeugten Overhead an Anfragevolumen abgewogen werden.

4.4 PPDNS

Lu und Tsudik [12] beschreiben das Modell Privacy Preserving Domain Name System (PPDNS), bei der das bestehende DNS umgebaut wird, so dass Anfragen gestellt werden, bei der der Nameserver nicht die Möglichkeit hat, die angefragte Domain zu bestimmen. Dazu wird das hierarchische Domain Name System durch eine flache verteile Hashtabellenstruktur (DHT) ersetzt. Als Beispiel wird das bereits existierende Projekt CoDoNS [16] verwendet, in dem jeder Knoten und jede Domain hat einen eindeutigen Wert (ID) aus dem Raum der Hashfunktion hat und der Knoten, dessen ID sich am wenigsten von dem Hash des Domainnamens unterscheidet, diesen permanent speichert.

Der grundlegende Ablauf ist dem der RangeQuery aus Ab-

doi: 10.2313/NET-2014-08-1_18

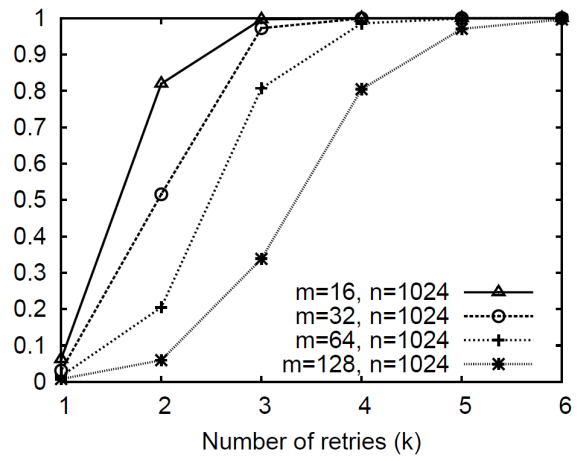


Abbildung 4: Wahrscheinlichkeit die angefragte Domain nach k Wiederholungen zu erkennen [12]

schnitt 4.3 ähnlich. Für eine DNS-Anfrage, bildet der Client einen Bereich, in dem der Hash der gesuchten Domain und $m - 1$ weitere Domains liegen. Um den korrekten Bereich der Hashwerte zu berechnen, in dem m Domains existieren muss die Dichte der Domains im Wertebereich der Hashfunktion bekannt sein. Diese bekommt er von seinem konfigurierten Nameserver, der diese aufgrund der Gleichverteilung der Hashfunktion mithilfe seiner eigenen Daten berechnen kann. Nachdem der korrekte Hashbereich gebildet wurde, wird dieser als Anfrage an den Nameserver gesendet. Liegt der angefragte Bereich im Cache des Nameservers, kann sofort geantwortet werden, ansonsten führt er einzelne Bereichsanfragen entsprechend der Struktur der DHT Knoten durch.

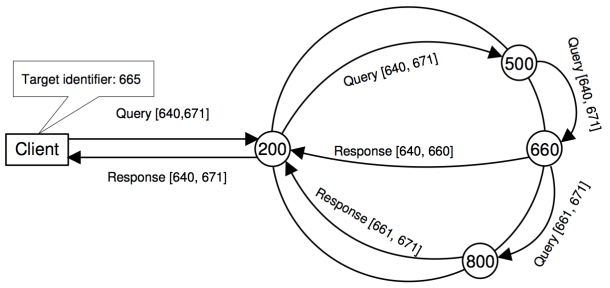


Abbildung 5: RangeQuery in einem DHT basiertem DNS [12]

In Abbildung 5 wird ein Ablauf einer Anfrage des Hashes 665 mit einer Hashfunktion durchgeführt, die einen Wertebreich von 1024 hat, mit einer Dichte von $\frac{1}{2}$ belegt ist und einem Sicherheitsparameter von $m = 16$ Domains pro Anfrage . Aus dem Anfrageziel 665 folgt eine Anfrage mit dem Bereich [640, 671], die an den konfigurierten Knoten mit der ID 200 gesendet wird. Da der Knoten diesen Bereich nicht im Cache liegen hat, wird die Anfrage weitergeleitet, bis alle Teilergebnisse des Bereichs vorliegen. Erst dann wird dem Client geantwortet und dieser kann aus dem Bereich die gewünschte Domain auswählen.

Durch RangeQuerys entsteht ein erheblicher Overhead, da auf jede Anfrage m Antworten versendet werden. Dadurch belegen RangeQuerys bei Verbindungen mit geringer Bandbreite einen Großteil der verfügbaren Übertragunsrate. Um diesen Overhead zu verkleinern, kann zusätzlich zu den RangeQuerys noch ein computational Private Information Retrieval (cPIR) Schema verwendet werden. Dafür wird der geforderte Bereich als eigene Datenbank gesehen, die in einzelne Unterbereiche unterteilt wird. Der Client kann nun einen bestimmten Index aus dieser Datenbank anfordern, ohne dass der Server erkennen kann welchen und ohne die komplette Datenbank zu senden. Lu und Tsudik [12] verwenden hierfür das cPIR Schema nach Gentry-Razam, welches die Ziele mit Arithmetik auf zyklischen Gruppen erreicht.

Lu und Tsudik [12] kommen zu dem Ergebnis, dass cPIR durch den großen Berechnungsaufwand auf aktueller Hardware nicht praktikabel einsetzbar ist. Bereits um einen Durchsatz von etwa 100 Anfragen pro Sekunde zu erreichen werden 100 Prozessoren benötigt. Im Gegensatz dazu ergeben die Simulationen mit reiner RangeQuery akzeptable Ergebnisse. So steigt die durchschnittliche Latenz einer RangeQuery innerhalb des PPDNS kaum an im Vergleich zu einer Anfrage mit lediglich einer einzelnen Domain. Da diese Simulationen jedoch nur auf Verbindungen innerhalb von Backbone Netzwerktopologien stammen, kann nicht davon ausgegangen werden, das dieses Ergebnis auch auf Endnutzer im privaten Bereich übertragbar ist, da hier die zugrundeliegenden Basislatenzen höher und die verfügbare Bandbreite sehr viel geringer ausfällt.

5. FAZIT

Nach Betrachtung möglicher Lösungsansätze zeigt sich, dass es kaum möglich ist, sich effizient vor den in Abschnitt 3.1.2 beschrieben Angriffe auf die Privatsphäre durch Betreiber von rekursiven Nameservern zu schützen, außer selbst die Namen iterativ aufzulösen, wodurch jedoch die autoritativen Nameserver mehr Informationen erhalten. Zwar existieren theoretische Modelle, in denen dieser Schutz gewährleistet werden kann, diese sind jedoch, wie in Abschnitt 4.4 gezeigt wurde, nicht ausreichend effizient um umgesetzt werden zu können. Des weiteren wird ein Umstieg auf ein solches System erschwert dadurch, dass ein kompletter, nicht kompatibler Umbau des Domain Name System nötig wäre. Auch die weitere Alternative der RangeQuerys innerhalb des normalen Domain Name Systems hat durch die Erhöhung des benötigten Traffics und Latenz kaum reelle Chance sich als Standard durchzusetzen.

Neben Effizienzgründen muss zusätzlich auch auf die Sinnhaftigkeit einer Einführung eines solchen Systems zur Verschleierung der DNS-Anfragen vor dem rekursiven Nameserver in Betracht gezogen werden. In vielen Netzwerksituationen, besonders im privaten Bereich, wird ein rekursiver Resolver des Internet Service Providers (ISP) verwendet. Das ist insofern problematisch, da der komplette Traffic durch das Netz des ISP muss. Dabei kann, sofern kein IP-Anonymisierungsdienst wie Tor verwendet wird, mindestens die IP bestimmt werden, mit der sich auf eine DNS-Anfrage folgend verbunden wird. Aus dieser kann unter Umständen per Reverse DNS Lookup ein Hostname gebildet werden und in Falle von RangeQuerys mit der ursprünglichen DNS-Anfrage verglichen werden. Auf diesen Weg kann der ISP

doi: 10.2313/NET-2014-08-1_18

die aufgelösten Namen zum Teil bestimmen obwohl diese in der originalen Anfrage verschleiert wurden.

Im Gegensatz zu den Lösungen, die auf Verschleierung der Anfrage vor den Serverbetreibern beruhen, finden Lösungsansätze für eine vertrauliche Datenübertragung zwischen zwei Parteien mehr und mehr Aufmerksamkeit und Attraktivität. Dabei hebt sich DNSCurve vor allem mit einem einfachen Protokoll und schneller Kryptographie auf Basis von elliptischen Kurven von Confidential DNS und t-DNS ab, welches durch die Verwendung von TCP und TLS als Basisprotokoll vor allem eine Erhöhung der Antwortzeiten bei der Kommunikation zwischen rekursiven und autoritativen Nameserver und der verbrauchten Ressourcen mit sich zieht. Auch lässt sich DNSCurve recht leicht in die bereits vorhandene Infrastuktur einbinden. Deshalb scheinen DNSCurve und DNSCrypt vielversprechende Kandidaten für einen standardisierten Einsatz in der Verschlüsselung von DNS-Traffic zu sein. So nutzt OpenDNS bereits seit Februar 2010 DNSCurve zur Verschlüsselung und Authentifikation von DNS-Traffic zwischen DNSCurve-fähigen autoritativen Nameservern [7]. Außerdem bietet es mit DNSCrypt bereits Software zur Verschlüsselung der Daten zwischen Stub Resolver und rekursive Nameserver an.

6. LITERATUR

[1] D. Atkins. Threat analysis of the domain name system (DNS). RFC 3833, RFC Editor, August 2004.

[2] D. J. Bernstein. Cryptography in NaCl. *Networking and Cryptography library*, 2009.

[3] D. J. Bernstein. DNSCurve: Usable security for DNS. http://dnscurve.org/, Jan 2011.

[4] S. Bortzmeyer. Possible solutions to DNS privacy issues. Internet-Draft draft-bortzmeyer-dnsop-privacy-sol-00, IETF Secretariat, December 2013.

[5] S. Bortzmeyer. DNS privacy considerations. Internet-Draft draft-bortzmeyer-dnsop-dns-privacy-02, IETF Secretariat, April 2014.

[6] M. Dempsky. DNSCurve: Link-level security for the domain name system. Internet-Draft draft-dempsky-dnscurve-01, IETF Secretariat, February 2010.

[7] M. Dempsky. OpenDNS adopts DNSCurve. http://blog.opendns.com/2010/02/23/opendns-dnscurve, February 2010.

[8] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, and D. Wessels. Starting TLS over DNS. Internet-Draft draft-hzhwm-start-tls-for-dns-00, IETF Secretariat, February 2014.

[9] S. Josefsson. DNSSEC Walker. http://josefsson.org/walker/, Nov 2005.

[10] P. Koch. Confidentiality aspects of DNS data, publication, and resolution. Internet-Draft draft-koch-perpass-dns-confidentiality-00, IETF Secretariat, November 2013.

[11] B. Laurie, G. Sisson, R. Arends, and D. Blacka. DNS security (DNSSEC) hashed authenticated denial of existence. RFC 5155, RFC Editor, March 2008.

[12] Y. Lu and G. Tsudik. Towards plugging privacy leaks in the domain name system. In *Peer-to-Peer Computing (P2P), 2010 IEEE Tenth International Conference on*, pages 1–10, Aug 2010.

[13] P. Mockapetris. Domain names - concepts and facilities. RFC 1034, RFC Editor, November 1987.

[14] P. Mockapetris. Domain names - implementation and specification. RFC 1035, RFC Editor, November 1987.

[15] OpenDNS. Introducing DNSCrypt. http://www.opendns.com/about/innovations/dnscrypt.

[16] V. Ramasubramanian and E. G. Sirer. Beehive: The design and implementation of a next generation name service for the internet. In *In ACM SIGCOMM*, 2004.

[17] W. Wijngaards. Confidential DNS. Internet-Draft draft-wijngaards-dnsop-confidentialdns-01, IETF Secretariat, March 2014.

[18] F. Zhao, Y. Hori, and K. Sakurai. Analysis of privacy disclosure in DNS query. In *Multimedia and Ubiquitous Engineering, 2007. MUE '07. International Conference on*, pages 952–957, April 2007.

[19] L. Zhu, Z. Hu, J. Heidemann, D. Wessels, A. Mankin, and N. Somaiya. T-DNS: Connection-oriented DNS to improve privacy and security. Technical Report ISI-TR-2014-688, USC/Information Sciences Institute, February 2014.

doi: 10.2313/NET-2014-08-1_18

Evil-Twin-Accesspoint

Andreas Huber
Betreuer: Marcel von Maltitz
Seminar Innovative Internettechnologien und Mobilkommunikation SS2014
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: hubera@in.tum.de

KURZFASSUNG

Diese Arbeit beschreibt den Angriff durch einen Evil-Twin-Accesspoint. Dazu wird im ersten Teil die exemplarische Vorgehensweise beim Durchführen dieses Angriffs erläutert. Aus der Perspektive des Angreifers sind die Rahmenbedingungen für einen erfolgreichen Angriff und die Möglichkeit des Angreifers gut zu erkennen. Im zweiten Teil der Arbeit werden Maßnahmen, die zum Erkennen und Verhindern dieses Angriffs vorgeschlagen wurden, auf Wirksamkeit und Durchführbarkeit verglichen. Hierbei wird auf die Möglichkeiten von Administratoren und Clients eingegangen. Zusätzlich werden Protokolle aufgezeigt, die Evil-Twin-Accesspoints verhindern sollen.

Schlüsselworte

Evil Twin, Evil-Twin Rogue Access Point, Angriff, Sicherheit WLAN

1. EINLEITUNG

Öffentlich zugängliche WLAN Netzwerke sind heute fast überall anzutreffen. Egal ob im Café, am Flughafen, in der Bibliothek oder im Restaurant, der Internetzugang über Funk ist selbstverständlich. Da es sich dabei oft um sehr kleine Netzwerke handelt, wird die Administration dieser Netze oft vernachlässigt. Auch aus Gründen des Komforts sind viele dieser Netzwerke nicht verschlüsselt. Da die einzige Information, die Netze für den Endanwender sichtbar voneinander unterscheidet, der Netzwerkname (ESSID) ist, ist es sehr einfach das öffentliche Netzwerk mit einem Laptop oder Smartphone nachzuahmen, indem mit dem mobilen Gerät ein Hotspot mit dem gleichen Namen, wie dem des öffentlichen Netzwerks, geöffnet wird. Es wird ein sogenannter „Evil-Twin-Accesspoint" gestartet. Ist dieser in der Nähe des Opfers, ist der Empfang des Evil-Twin-Netzes besser und das Opfer verbindet sich mit ihm. Das Opfer erkennt keinen Unterschied zwischen dem Hotspot des Angreifers und dem öffentlichen Netzwerk, da Anfragen an das Internet vom Angreifer über seine Internetverbindung weitergeleitet werden. Der Angreifer ist nun in der Lage, einen Man in the middle Angriff durchführen, er kann den Internetverkehr seines Opfers abhören, um beispielsweise Passwörter oder E-Mails mitzulesen und er kann den Internetverkehr gezielt beeinflussen, um das Opfer auf gefälschte Seiten umzuleiten, in denen Daten eingetragen werden im Glauben auf der offizielle Seite zu sein. In Wirklichkeit werden aber dem Angreifer die Daten ausgehändigt.

Der Aufbau eines Evil-Twin-Accesspoints im ersten Teil der Arbeit zeigt die einfache Durchführbarkeit dieses Angriffs und die Bedingungen, unter denen dieser Angriff möglich ist. Um diesen Angriff zu verhindern gibt es einige Ansätze, wie dieser erkannt und gestoppt werden kann. Es werden benutzer- und administratorseitige Möglichkeiten genannt. Mithilfe der Sicht aus Angreifer- und Verteidigerperspektive aus den ersten beiden Abschnitten werden daraufhin die Abwehrmöglichkeiten anhand wichtiger Kriterien für die Maßnahmen verglichen.

2. DURCHFÜHRUNG EINES EVIL-TWIN-AP-ANGRIFFS

Um die Funktionsweise und Wirksamkeit von Abwehrmaßnahmen gegen Evil-Twins beurteilen zu können, ist es hilfreich die Sicht eines Angreifers einzunehmen. Dazu wird im folgenden die Vorgehensweise zum Aufbau eines Evil-Twins erläutert.

2.1 Ausgangssituation

Der Evil-Twin-Accesspoint kann in Situationen eingesetzt werden, in der mindestens ein Rechner mit einem WLAN-Netzwerk verbunden ist, das keine Verschlüsselung benutzt, oder mit WEP, WPA oder WPA2 Personal gesichert ist, wobei der Schlüssel dem Angreifer bekannt ist. Der Beispielangriff arbeitet mit dem unverschlüsselten Netzwerk „Netzwerk1", mit dem der Benutzer verbunden ist. Abbildung 1 zeigt das Schema eines Evil-Twin-Angriffs, bei dem das Opfer eine Verbindung mit dem öffentlichen Netzwerk annimmt, in Wirklichkeit aber mit dem Evil-Twin verbunden ist, der die Anfragen in das öffentliche Netzwerk weiterleitet.

2.2 Ziel des Angriffs

Der Aufbau eines Evil-Twin-Accesspoints dient dem Angreifer dazu, einen Man in the middle Angriff auf das Opfer durchzuführen, bei dem private Daten des Opfers ausgelesen und der Internetverkehr umgeleitet werden kann.

2.3 Benötigte Tools

Um den Angriff durchführen zu können, benötigt man ein mobiles Gerät mit WLAN-Zugriff und einer weiteren Möglichkeit eine Verbindung mit dem Internet herzustellen. Dazu muss die WLAN-Karte des Gerätes einen Betrieb als gleichzeitiger Hotspot und Client erlauben, um den Internetverkehr des Opfers über eine Verbindung ins öffentliche Netz leiten zu können. Eine Alternative dazu ist die Einwahl des Angreifers in das Handynetz, oder dem Angreifer steht ein LAN zur Verfügung mit dem er die Nachrichten

doi: 10.2313/NET-2014-08-1_19

Abbildung 1: Aufbau des Angriffs

des Opfers ins Internet leitet. Von letzterem wird im folgenden Beispiel ausgegangen. Für diesen beispielhaften Angriff wurde ein Laptop mit Linux als Betriebssystem verwendet.

Das Programm `Aircrack-ng` wurde hauptsächlich entwickelt, um zu zeigen, wie einfach WEP-Netzwerke geknackt werden können. Es eignet sich auch als Universalwerkzeug, um Sicherheitslücken in WLAN-Netzen zu finden.[1] Im folgenden Angriff wird es zum Auslesen des Netzwerks und zum Erstellen des Hotspots benötigt. Zur Vergabe von gültigen IP-Adressen verwendet der Angriff den `ISC-DHCP-Server`[2]. Der Open Source DHCP Server kann mit einer Konfigurationsdatei angepasst werden. Das Routing der Internetverbindung in das WLAN-Netz erfolgt durch die Linux Kernel Programme `ifconfig` und `iptables`, die das Netzwerkinterface und die Firewall des Rechners konfigurieren. Weiterhin wird ein Programm zum Auslesen des Internetverkehrs benötigt. Der Beispielangriff nutzt hierzu `ngrep`[3], das anhand von regulären Ausdrücken Nachrichten aus dem Nachrichtenstrom filtert. Unter Ubuntu 13.10 können die vorgestellten Programme mit dem Befehl `apt-get install Aircrack-ng isc-dhcp-server ngrep` installiert werden.

2.4 Vorgehensweise

Die Vorgehensweise orientiert sich an [9]. Ein vorhandenes WLAN-Netz wird kopiert, indem ein WLAN-Netz mit gleichem Namen gestartet wird und die ursprüngliche Verbindung des Clients zum Accesspoint gestört wird.

2.4.1 Auslesen der WLAN-Infrastruktur

Zum Auslesen der WLAN-Infrastruktur wird die Programmsammlung `Aircrack-ng` benutzt. Ein Aufruf von `airmon-ng start wlan0 1` erzeugt ein Monitoring Device `mon0` auf Wifikanal 1. Mit dem Aufruf `airodump-ng mon0` werden die vorhandenen WLAN-Netze angezeigt. Der Aufruf liefert den Output aus Abbildung 2. Im oberen Teil des Outputs werden die gefundenen Netzwerke angezeigt, der untere Teil zeigt mit den Netzwerken verbundene Geräte. Von den gefundenen Netzwerken wird die MAC-Adresse des Routers (BS-

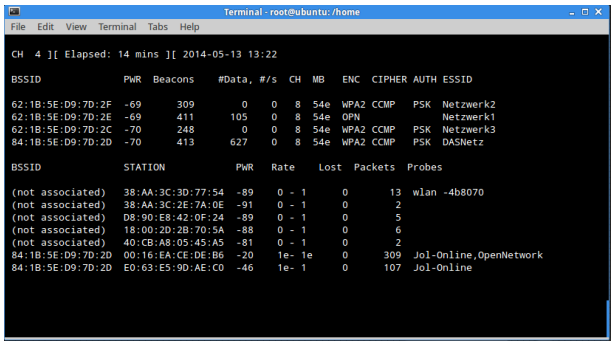


Abbildung 2: Ausgabe von Airodump-ng

SID), die Empfangsstärke (PWR), die Anzahl der empfangenen Initialisierungspakete (Beacons), die gesendeten Daten (#Data, #/s), der Kanal (CH), die Bandbreite des Netzwerks (MB), die Verschlüsselung (ENC, CIPHER, AUTH) und der Netzwerkname (ESSID) angezeigt. Zum Durchführen eines Angriffs wählt man das Opfer-Netz aus der Liste der vorhandenen Netze aus. Für die weiteren Schritte werden Netzwerkname, Router-MAC-Adresse und Kanal, also ESSID, BSSID und CH des Zielnetzwerkes benötigt. Hier wird das Netzwerk mit der ESSID „Netzwerk1", das auf Kanal 8 sendet und BSSID 62:1B:5E:D9:7D:2E besitzt, ausgewählt, da #Data zeigt, dass in dem Netzwerk Daten versendet werden, also mindestens ein Benutzer aktiv ist, und keine Verschlüsselung angewandt wird. Anschließend wird das Programm `airodump-ng` beendet und das Interface `mon0` mit den Befehlen `airmon-ng stop mon0` und `airmon-ng stop wlan0` geschlossen.

2.4.2 Konfiguration des DHCP-Servers

Zur Konfiguration des DHCP-Servers wird im Dateisystem die Datei `/etc/dhcp/dhcpd.conf` erstellt. Diese Konfigurationsdatei wird mit folgenden Werten beschrieben:

doi: 10.2313/NET-2014-08-1_19

Listing 1: Konfigurationsdatei des DHCP Servers

```
default-lease-time 600;
max-lease-time 7200;
subnet 192.168.2.128 netmask 255.255.255.128
{
   option subnet-mask 255.255.255.128;
   option broadcast-address 192.168.2.255;
   option domain-name-servers 8.8.8.8;
   option routers 192.168.2.129;
   range 192.168.2.130 192.168.2.140;
}
```

In der Konfigurationsdatei aus Listing 1 werden `default-lease-time` auf 600 Sekunden und `max-lease-time` auf 7200 Sekunden festgelegt. Diese Werte geben die Gültigkeitsdauer der DHCP-Konfiguration an. Nach Verstreichen der Default-lease-time bittet der Client um eine Verlängerung der Gültigkeit der IP-Konfiguration. War ein Client für die Dauer von Max-Lease-Time nicht verbunden, verfällt die IP-Adresse des Clients. Danach wird dem Client bei erneutem Verbinden eine andere IP-Adresse zugewiesen. Mit dem Befehl `subnet 192.168.2.128 netmask 255.255.255.128` wird das Subnet der Clients definiert, für das weitere Einstellungen vorgenommen werden. Die mit `option` angegebenen Werte überträgt der DHCP-Server an seine Clients. So erhalten die Clients Subnetzmaske, Broadcast-Adresse, die Adresse des Domain-Name-Servers und die Router-IP-Adresse. Diese wird später dem Angreifer-Laptop zugewiesen. Der angegebene DNS-Server an der IP-Adresse `8.8.8.8` ist ein öffentlicher DNS-Server von Google, an den die Clients ihre DNS-Anfragen richten werden. In dieser Konfiguration ist auch DNS Spoofing möglich. Dazu betreibt das Angreiferlaptop einen eigenen DNS-Server, um das Opfer auf Phishingseiten umzuleiten. In diesem Fall würde die IP-Adresse des Angreifer-Laptops als DNS Server angegeben. Die Zeile `range 192.168.2.130 192.168.2.140` legt den Bereich der verfügbaren IP-Adressen fest, die der Server vergeben kann.

2.4.3 Starten des Evil-Twin-AP

Jetzt kann der WLAN-Hotspot auf dem Laptop gestartet werden. Dazu wird erst mit `airmon-ng start wlan0 8` ein Aircrack-ng Monitoring Interface auf dem Kanal des Opfer-Netzwerks gestartet. Danach öffnet der Befehl `airbase-ng -e Netzwerk1 -c 8 -a 00:12:23:34:45:56 mon0` ein unverschlüsseltes WLAN-Netz mit der SSID „Netzwerk1" auf dem Kanal 8. Mit der Option -a wird dem Hotspot die willkürliche Mac-Adresse 00:12:23:34:45:56 zugewiesen. Das Programm läuft während des gesamten Angriffs. Abbildung 3 zeigt die Ausgabe des Programms `airbase-ng` nachdem sich ein Client mit dem Hotspot verbunden hat. Das Programm erstellt das Interface `at0`, auf dem die Nachrichten der Opfer ankommen. Verbindet sich ein Client mit dem Hotspot, wie beispielsweise in Abbildung 3 um 14:02:31 gezeigt, wird seine MAC-Adresse und das Netzwerk, mit dem er sich verbindet, angezeigt. Hier hat das Gerät die MAC-Adresse 64:20:0C:DF:2F:AC und verbindet sich mit dem „Netzwerk1", das vorher erstellt wurde. Der Ausdruck „reassociated" zeigt, das das Opfer-Gerät davon ausgeht im gleichen Netzwerk auf einen anderen Accesspoint zu wechseln. Das Opfer erkannte den Evil-Twin also nicht als solchen.

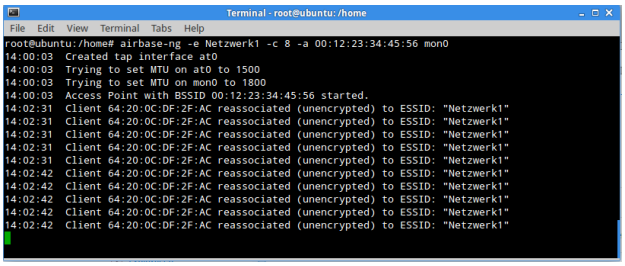


Abbildung 3: Ausgabe von airbase-ng nachdem sich ein Opfer verbunden hat.

2.4.4 Einrichten von Routing und DHCP-Server

In einem neuen Konsolenfenster wird nun das Routing so eingerichtet, dass die verbundenen Clients Internetzugang haben. Zuerst muss das Interface des Hotspots `at0` gestartet werden. Dies geschieht durch `ifconfig at0 up`. Daraufhin werden dem Interface mit `ifconfig at0 192.168.2.129 netmask 255.255.255.128` Router-IP-Adresse und Netzmaske des durch den DHCP-Server verwalteten Netzes zugewiesen. Der DHCP-Server wird durch den Befehl `dhcpd -cf /etc/dhcpd/dhcpd.conf` mit der vorher konfigurierten Datei `/etc/dhcpd/dhcpd.conf` gestartet. Um ein korrektes Laden der Konfigurationsdatei sicherzustellen, wird mit `service isc-dhcpd-server restart` der Server neu gestartet.

Die Linux Firewall besteht aus verschiedenen Tabellen, in denen jeweils bestimmte Nachrichten verändert, weitergeleitet oder gelöscht werden. Die Bearbeitung der Nachrichten in der Firewall erfolgt in Chains, in denen die Nachrichten vom einen Befehl zum nächsten gereicht werden. Die Konfiguration der Linux Firewall wird durch das Programm `iptables` durchgeführt.[5]

Die folgenden Befehle aus [9] konfigurieren die Firewall des Evil-Twins:

```
iptables --flush
iptables --table nat --flush
iptables --delete-chain
iptables --table nat --delete-chain
iptables --table nat --append POSTROUTING
         --out-interface eth0 -j MASQUERADE
iptables --append FORWARD -j ACCEPT
         --in-interface at0
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Mit den flush-Befehlen werden die Firewall-Regeln aus den Chains gelöscht, bevor mit delete-chain auch die vorhandenen Chains gelöscht werden. Beide Befehle werden sowohl für die Standardtabelle filter (ohne `--table`-Option), als auch für die Network Address Translation(NAT)-Tabelle durchgeführt. Der Befehl `iptables --table nat --append POSTROUTING --out-interface eth0 -j MASQUERADE` fügt der Chain `POSTROUTING` in der Chain der NAT-Tabelle eine Regel hinzu, die Nachrichten an das Interface `eth0` verändert, hinzu. Die hinzugefügte Regel `-j MASQUERADE` startet eine Network Address Translation für die dynamisch vergebenen IP-Adressen des Hotspots. So sieht das Netz, an dem der Angreifer mit dem Internet verbunden ist, nicht, dass mehrere Geräte auf das Netzwerk zugreifen und das Netzwerk kann den Angreifer nicht als solchen erkennen. Mit `iptables --`

doi: 10.2313/NET-2014-08-1_19

`append FORWARD -j ACCEPT --in-interface at0` wird der Chain der Standardtabelle (`filter`) der Firewall eine Regel hinzugefügt. Sie erlaubt allen Nachrichten aus dem Hotspot-Netz `at0` das Passieren der Firewall. Der Befehl `echo 1 > /proc/sys/net/ipv4/ip_forward` leitet eine Eins in die Datei `/proc/sys/net/ipv4/ip_forward`. Diese Konfiguration des Linux-Kernels erlaubt das Weiterleiten von IPv4-Nachrichten und das Angreifer-Laptop kann als Router arbeiten.

2.4.5 Auffinden und Stören von Client-Verbindungen

Der Befehl `airodump-ng --bssid 62:1B:5E:D9:7D:2E -c 8 mon0` mit MAC-Adresse und Kanal des Opfernetzwerks startet das Programm `airodump-ng`, das das Zielnetzwerk und seine Clients auflistet. Abbildung 4 zeigt die Ausgabe des Programms. In der Tabelle findet man die BSSID des potentiellen Opfers. Mit dieser kann im Folgenden eine Nachricht zur Unterbrechung der Verbindung vom Accesspoint an den Client gefälscht werden. Dazu benötigt man die MAC-Addresse des Opfers, die im unteren Teil der Ausgabe unter `STATION` abgelesen werden kann, hier 00:16:EA:CE:DE:B6 (Abbildung 4, mitte). Mit dem Befehl `aireplay-ng -0 0 -a 62:1B:5E:D9:7D:2E -c 00:16:EA:CE:DE:B6 -e Netzwerk1 --ignore-negative-one mon0` sendet der angreifende Rechner ständig Deauthentifizierungsnachrichten an den Client. Mit den Optionen `-a -c` und `-e` werden die MAC-Adressen von Accesspoint und Client und die ESSID des Netzwerks angegeben. Diese Informationen sind in den Deauthentifizierungsnachrichten enthalten und werden deshalb vom Programm als Eingabe benötigt. Abbildung 4 zeigt die Ausgabe des Programms, das in regelmäßigen Abständen Unterbrechungen sendet und die empfangenen Bestätigungen des Clients anzeigt. Die Verbindung des Opfers mit dem Originalnetzwerk bricht ab.

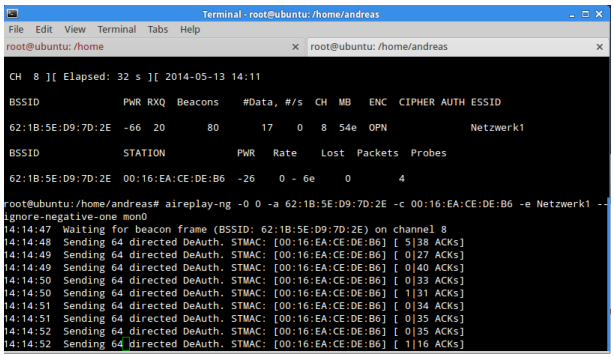


Abbildung 4: Ausgabe von airodump-ng und aireplay-ng

2.4.6 Opfer verbindet sich mit Angreifer

Da die Verbindung zum ursprünglichen Netzwerk abgebrochen ist, versucht der Client sich mit dem Netzwerk erneut zu verbinden. Dies geschieht je nach Gerät und Betriebssystem manuell oder automatisch. Da der Name des Originalnetzwerks mit dem des Angreiferhotspots identisch ist und eine Verbindung zum Originalnetzwerk durch das Programm `aireplay-ng` verhindert wird, verbindet sich das Opfer mit dem Netzwerk des Angreifers. Oft ist das Stören der Verbindung des Clients nicht notwendig. Eine entscheidende Rolle bei der (automatischen) Wahl des Accesspoints spielt die Empfangsstärke des Netzwerks. Ist das Angreiferlaptop dem Opfer näher als der Accesspoint, verbindet sich das Opfer-Gerät oft automatisch mit dem Angreifer, da das mobile Gerät aus Energiespargründen Accesspoints mit besserem Empfang bevorzugt.

2.4.7 Abgreifen und verändern von Daten

Nach einer erfolgreichen Verbindung läuft der gesamte Internetverkehr des Opfers über das Angreiferlaptop. Jetzt können beispielsweise die HTTP-Nachrichten des Opfers mit `ngrep -d at0 -t '^(GET|POST)' 'tcp and dst port 80 and src host 192.168.2.130'` abgegriffen werden. Die Option `-d` legt das Interface auf `at0` fest. Mit `-t` wird die Zeit der Nachricht mit angezeigt. Durch den regulären Ausdruck `'^(GET|POST)'` werden alle GET- und POST-Befehle des Opfers aufgezeichnet, die durch den Filter `tcp and dst port 80 and src host 192.168.2.130` gekommen sind, also vom Opfer mit der IP-Adresse 192.168.2.130 an den TCP-Port 80 gesendet wurden. Werden vertrauliche Nachrichten unverschlüsselt mit HTTP-GET oder HTTP-POST an den Server übertragen, so kann der Angreifer diese aufzeichnen und gelangt beispielsweise an Passwörter des Opfers. Durch Veränderung des regulären Ausdrucks und des Filters ist es auch möglich über HTTP, IMAP oder POP3 unverschlüsselt übertragene E-Mails des Opfers aufzuzeichnen.

Eine weitere Möglichkeit, die dieser Angriff bietet ist das Phishing von Online-Accounts. Dazu wird auf dem Angreifergerät ein DNS-Server, beispielsweise `bind` [4] gestartet. Der Domain Name Server-Eintrag in der DHCP-Serverkonfiguration aus Listing 1 wird auf die IP-Adresse des Angreiferlaptops `192.168.2.129` umgeändert. Der DNS-Server wird so konfiguriert, dass Anfragen zu der anzugreifenden Seite auf `192.168.2.129`, also auf das Angreiferlaptop geleitet werden, auf dem zusätzlich ein Webserver läuft, der eine zum Original möglichst identische Loginseite präsentiert, auf der das Opfer seine geheimen Daten eintippt und sie so dem Angreifer preisgibt.

Der vorgestellte Angriff ist ähnlich auch in Netzwerken möglich, die mit WEP verschlüsselt sind, vorausgesetzt der Angreifer kennt das Kennwort des Netzwerks. Das Auslesen der Netzinfrastruktur und die Konfiguration der Firewall bleiben gleich. Der einziger Unterschied ist der Aufruf von `airbase-ng` mit der Option `-w Schlüssel`, bei der der Netzwerkschlüssel angegeben wird. WPA-, WPA2-Personal- und WPA2-Enterprise-verschlüsselte Netzwerke können mit diesem Tool nicht gefälscht werden, allerdings ist es möglich ein unverschlüsseltes Netzwerk mit gleichem Namen zu erstellen. Wenn gleichzeitig das Originalnetzwerk mit `aireplay-ng` gestört wird, ist es möglich, dass sich Benutzer ohne weiteres Nachdenken mit dem Netzwerk mit gleichem Namen verbinden, falls dieses eine funktionstüchtige Internetverbindung bietet. Mit diesem Trick ist auch das Phishing des Netzwerkschlüssels möglich, wenn nämlich jeder Aufruf einer Internetseite auf eine Landing Page führt, die unabhängig von der aufgerufenen Seite auf jede Anfrage an das Web zurückgegeben wird, wo die Anmeldedaten des Nutzers abgefragt werden. Unvorsichtige Benutzer geben hier ihre Anmeldedaten ein und händigen sie so dem Angreifer aus.[9]

doi: 10.2313/NET-2014-08-1_19

2.5 Funktionsfähigkeit des Angriffs

Der Angriff, wie hier beschrieben wurde mit verschiedenen Geräten als Client getestet. Während ein PC mit Microsoft Windows 8.1 und ein Apple iPad mit iOS7 eine Verbindung zum Evil-Twin herstellten und je nach Empfangsqualität der Netze zwischen Originalnetzwerk und Evil-Twin wechselten, war es mit einem Android 4.4-Smartphone nicht möglich sich mit dem Evil-Twin zu verbinden. Dies zeigt, dass Evil-Twin-Angriffe schwer zu erkennen sind und auf verschiedenster Hard- und Software durchführbar sind.

3. MÖGLICHE ANGRIFFSABWEHR

Um das Abgreifen von Benutzer- und Firmendaten durch einen Evil-Twin-AP zu verhindern, gibt es sowohl auf der Seite des Netzes als auch auf der Seite des Benutzers Ansätze, wie Evil-Twins erkannt und entfernt werden können. Im folgenden werden jeweils zwei Möglichkeiten auf Administratorseite und auf Clientseite vorgestellt. Auch durch Änderung des WLAN-Protokolls und durch die Verwendung von verschlüsselten Protokollen in der Anwendungsebene kann ein Angriff eines Evil-Twins verhindert werden.

3.1 Administratorseitige Möglichkeiten

Es liegt im Interesse von Netzwerkadminstratoren das Netz gegen Angriffe von Evil-Twin-Accesspoints zu schützen, da Angreifer mit Evil-Twin-APs an vertrauliche Daten im Netzwerk gelangen können. Dazu überwacht der Administrator das Netzwerk, findet falsche APs und initiiert deren Abschaltung.

3.1.1 Funküberwachung

Bei der Funküberwachung werden die Funkfrequenzen von WLAN nach unberechtigten WLAN-Netzen abgesucht. Dazu werden die Daten der autorisierten Accesspoints im Netzwerk in einer Datenbank verwaltet. Es werden Daten, wie Netzwerkname, Funkkanal und MAC-Adresse des Accesspoints gesammelt und mit den Werten in der Datenbank verglichen. Startet ein Angreifer einen Evil-Twin des Originalnetzwerks, sieht das Messgerät das neue Netz und bemerkt, dass die MAC-Adresse des Accesspoints nicht bekannt ist und es sich um einen Angreifer handeln kann. Diese Maßnahme kann entweder in regelmäßigen Abständen manuell durch den Administrator durchgeführt werden, indem dieser mit einem mobilen Gerät den Sendebereich des WLAN-Netzwerks absucht, oder durch fest installierte Geräte, die eine ständige Überwachung durchführen. Durch Einschleusen von Broadcast-Paketen in das Netzwerk können zusätzlich Geräte gefunden werden, die nur sehr selten senden, wenn sie auf diese eingeschleusten Pakete reagieren und somit mit dem eigenen Netzwerk verbunden sind. Das Verhindern des Angriffs kann durch Finden und Entfernen des Angreifergeräts erfolgen, oder die Verbindungen des Angreifernetzwerks werden ähnlich wie in 2.4.5 durch Senden von Deauthentifizierungsnachrichten unterbrochen.[6]

3.1.2 Netzwerküberwachung

Bei der Netzwerküberwachung wird durch die Überwachung des Netzwerkverkehrs an einem zentralen Punkt, wie dem Router, versucht die Netzwerkinfrastruktur zu erkennen. Dazu werden die Daten der autorisierten Benutzer benötigt, ihre MAC- und IP-Adresse und weitere Charakteristiken, wie geöffnete Ports, müssen bekannt sein. Durch Einstellungen der Firewall am Router werden nur Pakete bekannter Adressen ins Internet geleitet. Eine andere Möglichkeit ist das aktive Absuchen. Mit dem Senden von Nachrichten an bestimmte Adressräume, die Evil-Twins nutzen könnten, versucht der Administrator Charakteristiken angeschlossener Geräte zu bekommen und vergleicht sie mit den Charakteristiken der erlaubten Geräte. Bei unbekannten Geräten wird von Angreifern ausgegangen. Sie werden vom Netzwerk ausgeschlossen.[6]

3.2 Clientseitige Möglichkeiten

Da der Client keinen Überblick über die Sicherheitsmaßnamen des Administrators im Netz hat und vor allem kleinere, öffentlich zugängliche Netzwerke oft schlecht geschützt sind, ist es für den Client wichtig, Möglichkeiten zu haben, festzustellen, ob man sich mit dem echten Netzwerk verbindet, oder ob die Verbindung über einen Evil-Twin-Accesspoint führt.

3.2.1 Zeitmessung

Wird davon ausgegangen, dass Evil-Twin-Accesspoints die Internetverbindung des öffentlichen WLAN Netzes nutzen, um selbst mit dem Internet verbunden zu sein und dieses WLAN-Netz ein älteres Netz nach dem Standard 802.11b/g ist, gibt es eine erfolgreiche Möglichkeit um auf der Clientseite Evil-Twin-APs erkennen zu können. Dazu wird die Zeit gemessen, die eine Nachricht im Netzwerk zu einem bestimmten Server braucht. Da eine Nachricht, die über einen Evil-Twin-AP gesendet wurde, nicht nur die Funkstrecke zwischen Benutzer und Accesspoint überwinden muss, sondern vom Benutzer zum Evil-Twin und von dort zum Accesspoint geleitet wird, ist die Zeit der Nachricht im Netzwerk entsprechend länger. Sind Charakteristiken des Netzwerks bekannt, kann die aktuelle Zeit zwischen zwei ankommenden Nachrichten genutzt werden, um sie mit den Durchschnittswerten im Netzwerk zu vergleichen. Da diese Charakteristik relativ stabil bei unterschiedlichen Nutzungsraten im Netzwerk verhält, ist der Einfluss der Anzahl der Benutzer im Netz gering. Lange Nachrichtenübertragungszeiten im Vergleich zum Durchschnitt des Netzwerks deuten auf eine Verbindung über einen Evil-Twin-AP hin. Ist die Charakteristik eines Netzwerks unbekannt, kann ein alternativer Algorithmus das Risiko eines Evil-Twin-Angriffs ohne Vorkenntnisse zum spezifischen Netz bewerten. Als Charakteristik hierzu dient der Quotient aus der Zeit zwischen zwei eingehenden Nachrichten vom Server und der Antwortzeit des nächsten Accesspoints, der entweder der Accesspoint im Originalnetzwerk ist, oder der Evil-Twin-AP. Je größer dieser Quotient ist, um so wahrscheinlicher ist ein Evil-Twin-AP, da dann die Nachricht zum Server im Vergleich zu einer Nachricht zum Accesspoint sehr lange dauert und somit die Strecke nach dem Accesspoint lang ist.[7]

Ein Problem dieses Ansatzes ist, dass er auf viele Annahmen aufbaut, die nicht immer bei Evil-Twin-APs zutreffen. Der Evil-Twin muss nicht unbedingt über WLAN und mit dem gleichen Netzwerk verbunden sein, er kann beispielsweise Zugriff auf Ethernet haben, oder mit dem Handynetz verbunden sein. Damit verändert sich die Nachrichtenübertragungszeit zwischen Evil-Twin und Server und der Evil-Twin ist nicht mehr als solcher identifizierbar. Es wird auch von einem Internetzugang mit sehr kurzen Reaktionszeiten aus-

doi: 10.2313/NET-2014-08-1_19

gegangen, die Reaktionszeiten zwischen AP und Server im Internet müssen kurz sein. Ist das nicht der Fall, wird die gemessene Antwortzeit des Servers zu lange und es wird fälschlicherweise ein Evil-Twin-Angriff angenommen. Der Ansatz in [7] setzt sich auch nur mit den älteren WLAN-Varianten 802.11b/g auseinander. Die kürzere Übertragungszeit von Nachrichten durch die höhere Bandbreite bei neueren WLAN-Varianten kann zu einem kleineren Unterschied der Nachrichtenübertragungszeiten führen und damit zum erschwerten Erkennen von Evil-Twins. Ist das Verhalten des Clients bekannt, kann der Evil-Twin den aufgerufenen Server simulieren und die Erkennung wird unmöglich.

3.2.2 Context Leashing [8]

Evil-Twin-Angriffe sind nicht nur an der Stelle möglich, an der das originale WLAN-Netz vorhanden ist, wenn ein Evil-Twin ein dem Opfer bekanntes Netzwerk an einem beliebigen anderen Ort öffnet, geht das Opfer vom Originalnetzwerk aus und verbindet sich mit dem Angreifer. Da der Angriff nicht im Originalnetz oder in dessen Nähe stattfindet, hat der Administrator des Netzes keine Möglichkeit diesen Angriff zu verhindern. Deshalb ist diese Maßnahme für die Clientseite wichtig. Durch das Vergleichen der vorhandenen Netze an der Stelle an der verbunden wird, kann dies verhindert werden. Dazu speichert das Gerät SSID und Empfangsstärke der umgebenden WLAN-Netzwerke beim ersten Verbinden mit dem WLAN-Netz. Beim erneuten Verbinden mit dem Netz wird die vorhandene WLAN-Infrastruktur mit den gespeicherten Werten verglichen. Wird dabei ein Unterschied der gespeicherten Netze zu den vorhandenen erkannt, wird von einem Evil-Twin ausgegangen. Um zu verhindern, dass ein Angreifer nicht nur das Opfer-Netzwerk nachahmt, sondern durch senden entsprechender Beacons alle Netze simuliert, kann zusätzlich zu den Namen der Netze deren Empfangsstärke ausgewertet werden. Dazu wird die Summe der Empfangsunterschiede von gespeichertem und aktuell gemessenem Netzwerk berechnet. Überschreitet die Summe einen Grenzwert, wird von einem Evil-Twin-AP ausgegangen.[8]

Dieses System macht den Aufwand eines erfolgreichen Evil-Twin-Angriffs an einer Stelle, an der das Netz nicht verfügbar ist, größer, allerdings kann der Angriff damit nicht ausgeschlossen werden. Wenn der Angreifer nicht nur die umgebenden Netze simuliert, sondern auch deren Sendestärke variiert und sie den gemessenen Werten anpasst, erkennt der Ansatz keinen Unterschied zur Originalinfrastruktur und eine Verbindung mit dem Evil-Twin wird hergestellt. Durch die Funktion moderner Smartphones und Rechner, die das einfache Öffnen eines Hotspots an beliebiger Stelle erlauben, um beispielsweise Daten auszutauschen, besteht bei diesem Ansatz außerdem die Gefahr, das echte Netze unrichtigerweise als Evil-Twins erkannt werden. Da diese mobilen Hotspots oft ein- und ausgeschaltet werden und zusätzlich häufig ihre Position ändern, ist es nicht sicher, ob sich die WLAN-Infrastruktur an einer Stelle bis zum nächsten Verbinden nicht ändert. Damit können originale Netze als Evil-Twin erkannt werden.

3.3 Angriffsverhinderung durch das Protokoll

Die Ansätze auf Client- und Administratorseite können Evil-Twin-Angriffe verhindern, ein vollkommener Schutz vor Evil-Twins ist durch diese Ansätze aber nicht sicher möglich, da alle Schutzmaßnahmen umgangen werden können. Diese Gefahr besteht auch unabhängig von der Verschlüsselung der WLAN-Verbindung. Auch wenn der Angriff für unverschlüsselte Netzwerke einfacher ist, als für verschlüsselte, kann ein Angreifer, der an den Schlüssel des Netzwerks gelangt ist, auch Evil-Twins von verschlüsselten Netzwerken zu erstellen, die nicht von den Original-Netzen zu unterscheiden sind. Selbst Netzwerke, die über WPA2 Enterprise geschützt sind, können mit entsprechendem Aufwand von einem Evil-Twin kopiert werden, wie in [10] gezeigt wird. Der Standard von WPA2 Enterprise 802.1X verwendet zur Authentifizierung der Clients das Extensible Authentication Protocol (EAP), das verschiedene Möglichkeiten der Authentifizierung bietet. Wird eines der weit verbreiteten Protokolle PEAP oder TTLS zur Authentifizierung verwendet, besteht die Gefahr eines Angriffs. Die Authentizität des Accesspoint-Betreibers wird durch X509-Zertifikate gewährleistet. Ist es dem Angreifer möglich sich ein Zertifikat der gleichen Zertifizierungsstelle, von der auch das Zertifikat des Originalnetzes stammt, ausstellen zu lassen, so vertraut der Client dem Angreifer-AP. Das häufig verwendete innere Authentifizierungsverfahren auf das der Angreifer jetzt Zugang hat ist MSCHAPv2, das mit etwas Aufwand geknackt werden kann. Gelingt dies, erkennt der Benutzer in dem Angreifer-AP einen gültigen Accesspoint.[11]

3.3.1 Protokoll mit Trust-On-First-Use

Der Schwachpunkt, der bei dem Angriff auf WPA2 Enterprise-Netze ausgenutzt wird, ist die Angreifbarkeit des EAP. Eine Möglichkeit diesen Schwachpunkt zu beheben, ist die Erweiterung des Protokolls um Trust-On-First-Use, das heißt, dass der Benutzer dem AP beim ersten Verbinden explizit vertraut, damit wird das Zertifikat des APs auf dem Client-Gerät gespeichert. Bei erneutem Verbinden wird das Zertifikat des APs mit dem bekannten Zertifikat verglichen. So wird sichergestellt, mit diesem AP bereits verbunden gewesen zu sein. Diese Funktionalität ist beispielsweise auch bei ssh umgesetzt. Durch diese Erweiterung des EAP-Protokolls, die auf dem häufig verwendeten Authentifizierungsverfahren TTLS basiert, und den Benutzer entscheiden lässt, ob der AP vertrauenswürdig ist, wird jeder Angriff, bei dem der Evil-Twin nicht beim ersten Verbinden mit dem Netzwerk aktiv ist, verhindert. Vor allem Netzwerke, mit denen häufiger eine Verbindung aufgebaut wird, werden so geschützt. Aber auch die Angreifbarkeit eines Benutzers der sich nur einmal mit dem Netzwerk verbindet, wird verringert, da ein Angriff, wie in Abschnitt 2 beschrieben, bei dem eine bestehende Verbindung unterbrochen wird, nicht mehr möglich ist. [8]

3.3.2 TLS/SSL in höheren Schichten

Vor dem Hintergrund, dass Nachrichten, die in das Internet gesendet werden, einen unbekannten Weg über zahlreiche Punkte, die den Netzwerkverkehr mitlesen und verändern können, nehmen, ist offensichtlich, dass kritische Daten immer verschlüsselt über das Internet übertragen werden sollten. Werden diese Daten aber bereits mit einem Verfahren, wie TLS übertragen, bei dem die Daten verschlüsselt werden und sich der Sender authentifiziert, ist ein Evil-Twin-AP keine Gefahr, da dieser weder Informationen mitlesen, noch Nachrichten verändern kann. Werden beispielsweise in einem öffentlichen Netzwerk nur Seiten aufgerufen, die über TLS/SSL gesichert sind, ist es für den Angreifer unmöglich

doi: 10.2313/NET-2014-08-1_19

die Nachrichten mitzulesen. Die Daten fließen zwar über den Evil-Twin-AP, aber der kann die verschlüsselten Daten nicht nutzen. Vor allem in öffentlichen WLAN-Netzen ist dies eine Methode, Mitleser auszuschließen. Um eine Verbindung über TLS zu ermöglichen, muss der Server, mit dem man sich verbinden möchte TLS unterstützen. Soll eine Verbindung zu einem Server aufgebaut werden, der kein TLS unterstützt, kann eine verschlüsselte VPN-Verbindung zu einem vertrauenswürdigen Proxy-Server aufgebaut werden. Von diesem Server werden die Nachrichten unverschlüsselt ins Internet gesendet, ohne dass ein Evil-Twin mitlesen kann.

3.4 Vergleich der Abwehrmöglichkeiten

Die Vorgestellten Abwehrmöglichkeiten von Evil-Twin-APs haben sehr verschiedene Eigenschaften. Entscheidende Dimensionen, die über den Erfolg der Abwehrmaßnahmen entscheiden, sind zum einen die Zuverlässigkeit der Maßnahme, außerdem ist die Einfachheit der Einführung einer Maßnahme entscheidend und zum dritten ist ein entscheidender Punkt, wie gut sich die Maßnahme auf große Netzwerke anwenden lässt.

3.4.1 Erkennungssicherheit/Zuverlässigkeit

Damit eine Angriffsabwehr sinnvoll ist, sollten möglichst alle Angriffe eines Evil-Twins erkannt werden. Die Abwehrstrategie sollte möglichst zuverlässig Evil-Twins erkennen und eine Verbindung mit ihnen verhindern.

Die Überwachung des Netzes auf Administratorseite ist nur möglich, wenn der Evil-Twin-AP in der Reichweite der Administrators aufgebaut wird. Ist der Angreifer nicht mit dem Netzwerk des Administrators verbunden, so hat dieser keine Chance den Angreifer anhand von Paketen im Netzwerk, wie in Abschnitt 3.1.2 beschrieben zu erkennen, da der Evil-Twin keine Pakete im Netzwerk sendet. Die Funküberwachung aus Abschnitt 3.1.1 erkennt Netzwerke, die im Bereich des Originalnetzes geöffnet werden, auch wenn keine Verbindung mit dem Originalnetz besteht. Allerdings hat sie keine Möglichkeit Angriffe zu verhindern, bei denen der AP an einem anderen Ort nachgebildet wird. Die administratorseitigen Ansätze können also vor allem das Netz und Daten im Netzwerk schützen. Da eine Verbindung vom Benutzer in ein anderes Netz nicht erkannt wird, können Benutzer nicht sicher geschützt werden. Ein weiterer Nachteil administratorseitiger Möglichkeiten ist, dass der Benutzer keinen Einblick auf die Arbeit des Administrators hat und nicht überprüfen kann welche Sicherheitsmaßnahmen dieser trifft und so nicht feststellen können, ob sich Netzwerk Evil-Twins befinden, die der Administrator nicht erkannte.

Die möglichen Maßnahmen der Clients erkennen auch nicht alle Angriffe sicher. Die Zeitmessung aus Abschnitt 3.2.1 baut auf die Annahme auf, dass der Evil-Twin über WLAN mit dem Original Netzwerk verbunden ist. Ist diese Voraussetzung nicht erfüllt, wird das Ergebnis des Algorithmus zum Erkennen des Evil-Twins sehr unzuverlässig, da beispielsweise eine schnellere Kabelverbindung die Dauer der Nachrichtenübertragung verkürzt. Die in Abschnitt 3.2.2 beschriebene Möglichkeit des Context Leashings bietet eine einfache Möglichkeit Evil-Twin Angriffe zu erschweren. Sie bieten zwar keinen absolut sicheren Schutz vor Angriffen, erschweren aber den Angriff vor allem an Orten, an denen andere oder keine WLAN-Netze vorhanden sind. Der Angreifer muss dann die Netz-Infrastruktur an dem Ort kennen, an dem das Gerät mit dem Netz verbunden war und diese Infrastruktur replizieren können. Wird eine Verbindung von Context Leashing auf Client-Seite und Funküberwachung auf Administratorseite angewendet, lässt sich ein relativ zuverlässiger Schutz vor Evil-Twin-Angriffen erreichen. Es gibt jedoch dann immer noch Möglichkeiten, wie erfolgreiche Evil-Twin Angriffe unternommen werden können, wenn beispielsweise der Evil-Twin die gesamte WLAN-Infrastruktur an anderer Stelle repliziert.

Durch ein Protokoll, das Trust-On-First-Use umsetzt (Abschnitt 3.3.1) lässt sich ein Angreifer am sichersten erkennen, vorausgesetzt es wurde bereits eine Verbindung aufgebaut und der Client kennt das Zertifikat des Netzes. Die einzige Möglichkeit, wie ein Evil-Twin nicht erkannt wird, ist dass er bereits bei der ersten Verbindung eine Kopie des APs präsentiert, mit der sich das Opfer verbindet. Bei der Verwendung von verschlüsselten Verbindungen in höheren Schichten als einzige Sicherheitsmaßnahme wird der Evil-Twin nicht erkannt. Der Angreifer, kann aber wegen der Verschlüsselung die gesendeten Nachrichten nicht lesen oder ändern. Wird jedoch nicht der gesamte Verkehr verschlüsselt, weil die Gegenstelle eine Verschlüsselung in höheren Schichten nicht unterstützt, kann der Evil-Twin die unverschlüsselte Nachricht mitlesen und so vertrauliche Daten erhalten.

3.4.2 Einfachheit der Umsetzung

Damit eine Abwehrmöglichkeit erfolgreich eingesetzt werden kann, ist es von Vorteil, wenn der Erfolg der Abwehrmaßnahme von möglichst wenigen abhängt. Der Vorteil der einseitigen Möglichkeiten zur Abwehr ist, dass die jeweils andere Seite keine Anpassungen durchführen muss, damit die Abwehr funktioniert. Eine Überwachung der Nachrichten im Netzwerk, wie in Abschnitt 3.1.2 ist relativ einfach, wenn sich die verbundenen Geräte nicht häufig ändern, es müssen nur die Adressen der berechtigten Geräte gespeichert und mit den Daten im Netzverkehr abgeglichen werden. Ändern sich allerdings die Clients häufig, wird der Aufwand die Liste der berechtigten Clients zu aktualisieren größer, so dass diese Abwehrmaßnahme unpraktikabel wird.

Das Überwachen des Funkverkehrs (Abschnitt 3.1.1) ist aufwändiger. Hier muss das Netzwerk entweder manuell überwacht werden, was wegen der häufigen Wiederholung im gesamten Netz sehr aufwendig ist, oder es müssen spezielle Geräte installiert und gewartet werden, die dies automatisch durchführen. Der größte Aufwand ist in diesem Fall die Installation und Einrichtung.

Die Möglichkeiten auf der Clientseite sind einfach in der Umsetzung. Die Zeitmessung in einem Netzwerk aus Abschnitt 3.2.1 kann unabhängig von der Verbindung als eigenständiges Programm umgesetzt werden, das Nachrichten ins Netzwerk sendet und die Zeit bis zur Antwort misst, und ist somit einfach in eine Netzwerkinfrastruktur zu integrieren. Ein Programm, das Context Leashing betreibt, benötigt, wie in Abschnitt 3.2.2 beschrieben, Informationen über vorhandene Netze und Empfangsstärke,das es einfach bekommen kann. Da aber keine zweite Seite involviert ist, ist die Umsetzung auch hier auch einfach möglich.

Möglichkeiten, die auf eine Änderung des Protokolls aufbau-

doi: 10.2313/NET-2014-08-1_19

en sind schwieriger umzusetzen, da beide Seiten die Neuerungen unterstützen müssen. Eine Erweiterung des EAP-Protokolls aus Abschnitt 3.3.1 muss von den Servern auf der Netzseite und von dem mobilen Gerät umgesetzt sein, um zu funktionieren. Da aber viele Hersteller von Hardware und Software beteiligt sind, gestaltet sich eine Einführung schwierig.

Auch für die Verwendung von Verschlüsselung in höheren Schichten muss diese von beiden Seiten unterstützt werden. Da aber heute TLS weit verbreitet ist alle mobilen Geräte über eine Implementierung von TLS verfügen, und auch Open-Source-Software für Server vorhanden und weit verbreitet ist, ist die in Abschnitt 3.3.2 beschriebene Verschlüsselung einfach umzusetzen.

3.4.3 Automatisierbarkeit und Skalierbarkeit

Damit die Abwehr von Evil-Twins erfolgreich ist, muss sie ständig und mit wenig manuellem Eingreifen durchführbar sein. Nur so können Fehler der Benutzer, die zu erfolgreichen Angriffen führen, verhindert werden.

Das Mitlesen und Auswerten von Nachrichten auf Administratrorseite aus Abschnitt 3.1.2 ist automatisch möglich. Da jedoch manuelle Einträge für jedes Gerät im Netz notwendig sind, ist diese Methode schlecht skalierbar.

Der manuelle Aufwand für eine Funküberwachung aus Abschnitt 3.1.1 ist groß, falls das Netz manuell mit einem mobilen Gerät abgesucht wird. Er beschränkt sich auf das aktualisieren der erlaubten Netze, falls für den ganzen Bereich des Netzes fest installierte Messgeräte aktiv sind, die die Funkfrequenzen überwachen, und so fremde Netze finden. In diesem Fall werden für größere Netze mehr Messgeräte benötigt, wodurch die Kosten für größere Netze steigen.

Durch das automatische Starten der Programme zum Erkennen von Evil-Twins lassen sich clientseitige Maßnahmen gut automatisieren. Da für die Messung der Antwortzeit die Netzgröße unerheblich ist, ist die Skalierbarkeit für die Methode aus Abschnitt 3.2.1 gut.

Auch das Context Leashing aus Abschnitt 3.2.2 kann unabhängig von der Größe des Netzes durchgeführt werden. Ein Problem könnte dabei sein, dass große Netze viele Nachbarnetze haben und somit das richtige Netz oft für einen Evil-Twin gehalten wird. Da hier manuelles Abschätzen des Risikos notwendig ist, hat in diesem Fall die Automatisierung Grenzen.

Wird das Protokoll zum Schutz vor Evil-Twins angepasst, wie in Abschnitt 3.3.1, spielt die Größe des Netzes keine Rolle, die manuellen Eingriffe beschränken sich auf das Bestätigen des Vertrauens zum Netz.

Bei der Verwendung von TLS, wie in Abschnitt 3.3.2 beschrieben, sind keine Eingriffe in das Netzwerk nötig. Die Größe des Netzwerks spielt deshalb keine Rolle.

4. ZUSAMMENFASSUNG

In dieser Arbeit wurde das Aufbauen eines Evil-Twin-AP gezeigt. Es wird deutlich, dass der Angriff mit wenigen frei verfügbaren Tools in kurzer Zeit durchgeführt werden kann. Die Ziele des Angriffs können vielfältig sein. Der Angriff kann genutzt werden, um Nachrichten des Opfers mitzulesen und zu verändern. Durch das Betreiben eines DNS-Servers kann das Opfer auf Phishingseiten umgeleitet werden, auf denen es ungewollt Passwörter preisgibt. Vor diesem Angriff schützt keine Verschlüsselungsart völlig zuverlässig. Nicht nur unverschlüsselte und WEP oder WPA/WPA2 verschlüsselte Netzwerke mit gemeinsamen Schlüssel können kopiert werden, auch Netzwerke die WPA2 Enterprise nutzen, können mit entsprechendem Aufwand kopiert werden. Zum Verhindern der Evil-Twins gibt es Ansätze auf Administratorseite, Benutzerseite und Ansätze, die das Protokoll der Authentifizierung so verändern, dass Evil-Twins erkannt werden. Der Administrator kann durch Überwachen der Nachrichten im Netz unbekannte Geräte erkennen, und sie stoppen, oder durch Überwachung des Funkverkehrs unberechtigte Accesspoints erkennen. Der Client kann aus der Dauer der Nachrichtenübertragung Rückschlüsse auf die Echtheit des APs ziehen. Durch Beobachten der umgebenden Netzinfrastruktur kann der Client seinen Standort abschätzen und so eine Verbindung zum Accesspoint nur am richtigen Ort aufbauen, wodurch der Angriff an anderer Stelle vermieden wird. Die Ansätze haben unterschiedliche Einsatzbereiche. Maßnahmen, die nur von Benutzer oder Administratoren durchgeführt werden, können die Angriffe nicht verhindern, sie erschweren diese allerdings. Vor allem das Context Leashing auf der Clientseite und die Funküberwachung durch den Administrator sind wirkungsvolle Maßnahmen, die gemeinsam angewendet Angriffe deutlich erschweren. Durch Erweiterung des EAP-Protokolls um Trust-On-First-Use wird die Möglichkeit eines Evil-Twin-Angriffs in WPA2-Netzwerken weiter verringert. Weil aber eine Verschlüsselung der Daten in höheren Schichten heute bereits sehr weit verbreitet ist und weiter zunimmt, verliert der Evil-Twin-Angriff an Relevanz, da durch die Verwendung von TLS das Mithören und Verändern der Nachrichten verhindert wird und das nicht nur im lokalen Netzwerk, sondern sondern auch durch das gesamte Internet bis zum Empfänger der Nachricht. Dann kann man auch im Cafe das öffentliche WLAN nutzen, ohne Angst haben zu müssen, abgehört zu werden.

5. LITERATUR

[1] *Aircrack-ng Home Page*, `http://www.aircrack-ng.org/`

[2] *ISC DHCP SERVER*, `http://www.isc.org/downloads/dhcp/`

[3] *ngrep - network grep*, `http://ngrep.sourceforge.net/`

[4] *ISC BIND DNS SERVER*, `http://www.isc.org/downloads/bind/`

[5] *man pages of iptables*, `http://ipset.netfilter.org/iptables.man.html`

[6] Airdefence, *TIRED OF ROGUES? Solutions for Detecting and Eliminating Rogue Wireless Networks*, White paper, `http://goo.gl/uuZBEH`

[7] Y. Song and C. Yang and G. Gu: *Who Is Peeping at Your Passwords at Starbucks? – To Catch an Evil Twin Access Point*, In Proceedings of the 40th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'10), Juni 2010

[8] H. Gonzales, K. Bauer, J. Lindqvist, D. McCoy, D. Sicker: *Practical Defenses for Evil Twin Attacks in*

802.11, In IEEE Globecom Communications and Information Security Symposium, Miami, FL, Dezember 2010

[9] *Video Tutorial zum Aufbau eines Evil-Twin-Angriffs* `http://vimeo.com/34309678`

[10] Wright, J. Antoniewicz, B.*PEAP Shmoocon 2008* `http://goo.gl/VZoyks`

[11] Antoniewicz, B. *802.11 Attacks*, 2010

doi: 10.2313/NET-2014-08-1_19

Analyse zur Sicherheit von Open-Source Software

Marcus Rogowsky
Betreuer: Dr. Heiko Niedermayer
Seminar Innovative Internettechnologien und Mobilkommunikation, SS 2014
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: m.rogowsky@tum.de

KURZFASSUNG

Open-Source Software ersetzt heute in Firmen, Behörden und privatem Umfeld immer mehr kommerzielle Closed-Source Software und ist durch die weite Verbreitung zu einem attraktiven Angriffsziel für Hacker geworden. Diese Entwicklung hat zu einer Debatte über die Sicherheit von Open-Source Software im Vergleich zu Closed-Source Software geführt. Es ist allerdings schwierig die unterschiedlichen Einflüsse der beiden Strategien auf die Software-Sicherheit zu modellieren. Die veröffentlichten Modelle betrachten entweder nicht den Unterschied zwischen Open-Source Software und Closed-Source Software oder liefern Ergebnisse, die sich nicht mit den Ergebnissen empirischer Studien über die Software-Sicherheit von großen Open-Source Projekten decken. Durch die Berichterstattung über den Heartbleed Bug in der OpenSSL Bibliothek ist diese Problematik in das Blickfeld der öffentliche Wahrnehmung gerückt.

Schlüsselworte

Open-Source Software, Closed-Source Software, Security, Heartbleed

1. EINLEITUNG

Der Einsatz von Open-Source Software hat in den letzten 20 Jahren auf allen Computer Plattformen stark zugenommen. Für Server, Desktop PCs und Smartphones werden Betriebssysteme und Software angeboten, die frei verwendet werden können und deren Code offen für jeden einzusehen ist. Damit unterscheidet sie sich von proprietärer Software, bei der der Code nur dem Entwickler bekannt ist und die vor dem Einsatz meist bezahlt oder lizenziert werden muss.

Durch die weite Verbreitung von Open-Source Software im behördlichen, geschäftlichen und privaten Umfeld wird diese auch ein attraktives Ziel für Angreifer, die Schwachstellen ausnutzen und damit Zugriff auf vertrauliche Daten bekommen oder Schäden verursachen können. Vor allem bei Systemen, die mit offenen Netzwerken wie dem Internet verbunden sind, muss die Sicherheit der verwendeten Software gewährleistet sein.

Da Open-Source Software heute die Basis für viele geschäftskritische Bereiche und kritischer Infrastruktur bildet, müssen hier überprüfbare Standards und Methoden zur Feststellung der Sicherheit von Software entwickelt werden. Ein wichtiger Aspekt ist dabei der Vergleich der Sicherheit von Open-Source Software mit der von Closed-Source Software und die Abwägung der Vor- und Nachteile beider Strategien.

In den beiden nachfolgenden Abschnitten wird eine Einführung in das Thema Open-Source Software und die Debatte über deren Sicherheit gegeben. Im Abschnitt 4 werden allgemeine Modelle zur Software-Sicherheit vorgestellt und ein Modell betrachtet, dass zwischen Open-Source Software und Closed-Source Software unterscheidet. Die Aussagen des Modells werden dann anhand einer empirischen Studie überprüft. Im darauffolgenden Abschnitt 5.2 wird ein Einblick gegeben, wie und in welchem Umfang Open-Source Software in Firmen eingesetzt wird. Da die Berichterstattung über den Heartbleed Bug die Wahrnehmung der Öffentlichkeit zu dem Thema Sicherheit von Open-Source Software grundlegend verändert hat, wird im letzten Abschnitt erklärt wie dieser Bug aufgetreten ist und welche Auswirkungen er gehabt hat.

2. DEFINITION VON OPEN-SOURCE SOFTWARE

Der Begriff Open-Source Software wurde mit der Gründung der Open Source Initiative 1998 eingeführt. Grundsätzlich ist eine Software als Open-Source Software zu verstehen, wenn der Quellcode für jeden zu Verfügung steht und die Software frei genutzt, kopiert und verteilt werden darf. Wie dabei der Entwicklungsprozess der Software genau abläuft und wie Codeänderungen in die originale Version aufgenommen werden, kann sich bei verschiedenen Projekten stark unterscheiden.

Eric S. Raymond, Mitbegründer der Open Source Initiative, beschrieb in seinem Essay „Die Kathedrale und der Basar“ [4] 1999 die zwei weitverbreitetsten Methoden als „Basar“ und „Kathedrale“. Bei der als „Kathedrale“ bezeichneten Methode wird der Entwicklungsprozess von einer kleinen Gruppe von Entwicklern kontrolliert, die die offizielle Version der Software erstellen und den Quellcode für jede veröffentlichte Version zur Verfügung stellen. Zwischen den einzelnen Veröffentlichungen kann die Öffentlichkeit nur indirekten Einfluss auf die Entwicklung nehmen. Dagegen setzt die als „Basar“ bezeichnete Methode auf die Zusammenarbeit mit der Öffentlichkeit über das Internet und erlaubt es jedem den Code in der Versionsverwaltung zu ändern oder neuen Code hinzuzufügen. Der aktuelle Quellcode des Projekts steht somit der Öffentlichkeit zu jedem Zeitpunkt in der Entwicklung zur Verfügung und Änderungen laufen sofort durch einen Peer-Review Prozess.

doi: 10.2313/NET-2014-08-1_20

3. DEBATTE OPEN-SOURCE SOFTWARE VS. CLOSED-SOURCE SOFTWARE

Zwischen den beiden Lagern der Open-Source und Closed-Source Softwareentwickler findet eine Debatte darüber statt, welches Modell der Entwicklung mehr Sicherheit bietet. Einig sind sich beide Seiten darin, dass in offenem Quellcode Bugs schneller gefunden werden und damit mehr Sicherheitslücken aufgedeckt werden. Die Auswirkungen auf die Sicherheit der Software werden jedoch unterschiedlich bewertet.

Befürworter des Open-Source Modells führen an, dass durch den Peer-Review Prozess den die Software durchlaufen muss die Codequalität generell höher ist und weniger Sicherheitslücken in der veröffentlichten Version vorhanden sind. Eric S. Raymond schrieb dazu in seinem Essay [4] „Given enough eyeballs, all bugs are shallow.“ Allerdings spielt bei der Beseitigung der gefundenen Bugs der verwendetete Entwicklungsprozess eine entscheidende Rolle. Bei einem dem „Basar“-Modell folgendem Entwicklungsprozess kann jeder der Schwachstellen findet diese direkt im originalen Code beseitigen. Wird dagegen das „Kathedralen“-Modell verwendet, kann man als Außenstehender nur die Entwickler des Projekts benachrichtigen, die dann über das weitere Vorgehen entscheiden. Ein entscheidender Vorteil der Open-Source Methode ist, dass man als Anwender die Funktionalität der Software selbst überprüfen kann und nicht darauf vertrauen muss, dass keine versteckten Hintertüren von den Entwicklern eingebaut wurden.

Dagegen argumentieren die Befürworter des Closed-Source Modells, dass das bloße Bereitstellen des Quellcodes keine Garantie für Sicherheit darstellt. Elias Levy merkt in seinem Artikel „Wide Open Source“ [5] 2000 an: „Sure, the source code is available. But is anyone reading it?“ Seiner Erfahrung nach lesen Endanwender der Software den dazugehörigen Code überhaupt nicht oder sind nicht qualifiziert genug, um Sicherheitslücken zu finden. Viele potenzielle Reviewer gehen zudem davon aus, dass der Quellcode von anderen bereits überprüft wurde. Dabei ist das Projekt oft so komplex, dass es nur im Rahmen professioneller Audits untersucht werden kann, wie es zum Beispiel bei der Verschlüsselungssoftware Truecrypt der Fall ist. In Softwarefirmen gäbe es außerdem feste Vorgaben an die Entwickler und eine klar strukturierte Organisation. Bevor die Software veröffentlicht wird, wird sie von speziell geschultem Personal in der Qualitätssicherungsabteilung überprüft. Und durch die Geheimhaltung des Quellcodes würde ein zusätzlicher Schutz geschaffen, der es für Angreifer schwieriger macht Schwachstellen zu finden und diese würden sich deshalb einfachere Ziele vornehmen.

4. MODELLE ZUR SOFTWARE-SICHERHEIT

Die öffentliche Debatte über die Sicherheit von Open-Source Software ist geprägt von verschiedenen Vorurteilen gegenüber der jeweiligen Entwicklungsmethode. Um allerdings den Einfluss von Open-Source auf die Sicherheit untersuchen zu können, braucht man Modelle, die Vorhersagen darüber machen können, wie verschiedene Faktoren die Sicherheit der veröffentlichten Software bestimmen. Diese Modelle sollten dann Hinweise darauf geben aus welchen Gründen Fehler auftreten, und wie diese in Zukunft vermieden werden können.

4.1 Modelle für Software allgemein

Schryen und Kadura [2] geben einen Überblick über die wichtigsten verwendeten Modelle und untersuchen, worin ihre Schwächen liegen. Die Grundlage dieser Modelle basiert auf der Methode von Littlewood et al. und Kimura, die die Zuverlässigkeitstheorie auf Software-Sicherheit anwenden. Erfolgreiche Angriffe auf eine Software werden dabei als Ereignisse betrachtet, die aufgrund von Sicherheitslücken auftreten. Für die Zeit bis zum nächsten Eindringen in das System wird eine Exponentialverteilung angenommen, wobei die Autoren keine Aussagen über den Faktor λ für die Anzahl der erwarteten Ereignisse pro Zeitintervall machen. Um diesen Faktor genauer zu spezifizieren, beobachteten Jonsson und Olovsson das Verhalten von Angreifern auf einem verteilten UNIX-System und wie schwer es für diese war in das System einzudringen. Mit ihren Ergebnissen stellten sie die Hypothese auf, dass das Auftreten von Sicherheitsverstößen in drei Phasen unterteilt werden kann: Die Lernphase, die Standard-Angriffsphase und die innovative Angriffsphase. Während der Standard-Angriffsphase ist die Wahrscheinlichkeit, dass ein Angriff erfolgreich ist deutlich höher als während den anderen beiden Phasen. Innerhalb der jeweiligen Phasen folgen die Zeiten zwischen erfolgreichen Angriffen der Exponentialverteilung was die Annahme unterstützt, dass das Zuverlässigkeitsmodell auf Software-Sicherheit angewendet werden kann. Summiert man alle bisher gefundenen Sicherheitslücken auf bilden die drei Phasen eine „S“ Kurve (Abbildung 1).

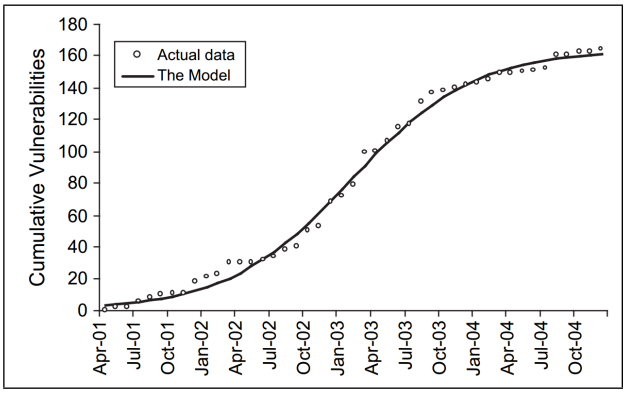


Abbildung 1: [6] S-Kurve für Linux 7.1

Alhazmi et al. [6] erklären diesen Verlauf damit, dass die Rate mit der Schwachstellen gefunden werden linear von zwei Faktoren abhängt: Die Verbreitung der Software und den noch zu finden Schwachstellen in der Software. Basierend auf diesen Annahmen stellten sie eine Gleichung auf, mit der sich anhand des bisherigen Verlaufs der Anzahl an gefunden Schwachstellen die Gesamtanzahl an Schwachstellen in der Software vorhersagen lassen sollte. Beim späteren Vergleich dieser Vorhersagen mit der Anzahl an tatsächlich gefundenen Schwachstellen stellte sich die vorhergesagte Anzahl als deutlich zu niedrig heraus. Mögliche Erklärungen dafür könnten sein, dass durch Patches später zusätzliche Sicherheitslücken in die Software eingefügt wurden, oder dass der Aufwand der betrieben wird um Sicher-

doi: 10.2313/NET-2014-08-1_20

heitslücken zu finden nicht linear mit der Zeit verläuft, zum Beispiel wenn die Anzahl an Reviewern über die Zeit schwankt.

4.2 Modell mit Unterscheidung von Open-Source Software und Closed-Source Software

Das Problem mit den bisher genannten Modellen ist, dass sie den Unterschied zwischen Open-Source und Closed-Source nicht berücksichtigen. Außerdem wird laut Schryen und Kadura [2] die Entwicklung und Überprüfung von Modellen dadurch erschwert, dass zuverlässige Daten zur Software-Sicherheit nur schwer zugänglich oder von zu schlechter Qualität sind.

Um die Unterschiede zwischen Open-Source und Closed-Source berücksichtigen zu können, schlagen Härtig et al. [1] ein Modell vor, dass die Reviewer in zwei Gruppen aufteilt. In der einen Gruppe sind die Angreifer, die Fehler finden wollen, um sie auszunutzen und in der anderen Gruppe die Verteidiger, die Fehler finden wollen, um sie zu beheben und die Qualität der Software zu verbessern. Die Software wird aufgeteilt in einzelne Einheiten, die entweder einen Fehler aufweisen oder fehlerfrei sind und beiden Gruppen jeweils eine Wahrscheinlichkeit zugeordnet Fehler zu finden. Im Fall von Open-Source Software bekommen beide Gruppen die gleiche Wahrscheinlichkeit Fehler zu finden, im Fall von Closed-Source Software wird die Wahrscheinlichkeit für die Angreifer um einen Faktor α verringert.

Die Fehlersuchphase wird als schrittweises Ziehen von Kugeln mit Zurücklegen modelliert, wobei Angreifer und Verteidiger aus unterschiedlichen Urnen ziehen, um die jeweiligen Wahrscheinlichkeiten abzubilden. Für beide Gruppen gelten unterschiedliche Gewinnbedingungen: Die Angreifer können einen beliebigen Fehler finden, der von den Verteidigern noch nicht gefunden wurde, um zu gewinnen. Die Verteidiger müssen jedoch jeden Fehler vor den Angreifern finden, um zu gewinnen. Wenn die Angreifer Fehler A finden, ist es irrelevant ob die Verteidiger Fehler B oder C gefunden haben, nur ob sie Fehler A gefunden haben oder nicht, ist entscheidend. Ziehen ein Angreifer und ein Verteidiger im gleichen Schritt denselben Fehler gewinnt der Angreifer, da er diesen sofort ausnutzen kann, ohne auf die Qualität seiner Schadsoftware zu achten. Der Verteidiger muss dagegen den Fehler erst beheben, ohne unerwünschte Nebeneffekte in der Software zu verursachen und die Endanwender dann den Patch herunterladen und installieren.

Um mit ihrem Modell Aussagen über den Unterschied zwischen Open-Source und Closed-Source Projekten zu machen, wenden sie dieses auf ein fiktives Softwareprojekt an, das typisches Fehlerverhalten aufweist und berechnen damit wie viele Verteidiger benötigt werden, um mit einer vorgegebenen Wahrscheinlichkeit alle Fehler vor den Angreifern zu finden. Mit einer Größe von 1 Millionen Zeilen Code und üblichen Werten von 0,3 Fehlern pro 1.000 Zeilen von denen 5% sicherheitsrelevant sind ergibt das 15 relevante Fehler im gesamten Projekt. Die Anzahl an Angreifern wird auf 500 geschätzt, da keine aussagekräftigen Studien dazu gefunden werden konnten. Für die Wahrscheinlichkeit, dass ein Verteidiger in einem Schritt einen Fehler findet, werden Werte von $q = 0,001\%$ bis $q = 0,005\%$ angenommen. Für die Angreifer werden bei Open-Source Software die gleichen Werte und bei Closed-Source Software um den Faktor $\alpha = 10$ kleinere Werte verwendet.

Tabelle 1: [1]
Anzahl benötigter Verteidiger für $p_w = 0.6$

$p_w = 0.6$	$q = 0.001\%$	$q = 0.002\%$	$q = 0.005\%$
$\alpha = 10$	1455	1466	1500
$\alpha = 1$	15630	17133	26245

Tabelle 2: [1]
Anzahl benötigter Verteidiger für $p_w = 0.9$

$p_w = 0.9$	$q = 0.001\%$	$q = 0.002\%$	$q = 0.005\%$
$\alpha = 10$	7360	7654	8774
$\alpha = 1$	124176	unmöglich	unmöglich

Zunächst wird eine sehr niedrige Gewinnwahrscheinlichkeit von $p_w = 0.6$ für die Verteidiger gewählt, wozu man laut Modell bereits etwa 15.000 bis 26.000 Verteidiger im Fall von Open-Source Software braucht, während man bei Closed-Source Software mit nur etwa 1.500 Verteidigern auskommt (Tabelle 1).

Für eine realistischere Betrachtung des Modells wird ein Wert von $p_w = 0.9$ für die Gewinnwahrscheinlichkeit der Verteidiger gewählt. Mit diesem Wert benötigt man mit Open-Source Software allerdings eine unrealistische hohe Anzahl an Verteidigern von über 100.000 oder es ist unmöglich diesen Wert zu erreichen, während bei Closed-Source Software etwa 8.000 Verteidiger ausreichen (Tabelle 2).

Die Aussage des Modells ist damit, dass egal wie viele Verteidiger es gibt, die Angreifer in einem von drei Fällen gewinnen werden. Die Autoren schließen daraus, dass man Angreifer nur mit Geheimhaltung des Quellcodes effektiv abwehren kann.

5. STUDIEN

Die Aussage des Modells aus Abschnitt 4.2 scheint nicht mit der objektiven Wahrnehmung zur Sicherheit von Open-Source Software übereinzustimmen. Sollten sich die Aussagen des Modells bestätigen, wäre der Einsatz von Open-Source Software in kritischer Infrastruktur grob fahrlässig, da diese praktisch nicht gegen Angreifer zu verteidigen wäre. Außerdem findet in den Modellen keine Unterscheidung von schweren Fehlern zu weniger schweren Fehlern statt, obwohl eine schwere Sicherheitslücke durch die Angreifer Administratorrechte auf dem Zielcomputer bekommen können schlimmer ist als 10 weniger schwere Sicherheitslücken, die nur Lesezugriff auf bestimmte Daten ermöglicht.

5.1 Empirische Studie

Um diese Fragen weiter zu untersuchen erstellte Schryen [3] eine empirische Studie, die die Sicherheit von 17 Softwareprojekten vergleicht. Für jedes Anwendungsgebiet wurden mindestens eine Open-Source Software und eine Closed-Source Software ausgewählt, die weitverbreitet sind und den gleichen Zweck erfüllen. Die Daten für die Analyse

doi: 10.2313/NET-2014-08-1_20

stammen dafür aus der National Vulnerability Database (NVD) des NIST, der größten Datenbank für Softwarebugs und sind nach dem Common Vulnerability and Exposures (CVE) Standard abrufbar. Die CVE Liste enthält eindeutige Bezeichner für jede Sicherheitslücke und wird von der MITRE Corporation in Zusammenarbeit mit Sicherheitsexperten und Softwareherstellern verwaltet.

Tabelle 3: [3] Daten über Schwachstellen (2011)

Software	Entwicklung	#Vuln	MTBVD	EASZ
IE 7	closed	74	13.29	linear
Firefox	open	167	5.16	linear
MS Outlook Express 6	closed	23	120.73	linear
Thunderbird 1	open	110	13.79	nicht linear
IIS 5	closed	83	40.90	nicht linear
Apache 2	open	80	40.63	linear
MS Office 2003	closed	99	19.22	nicht linear
OpenOffice 2	open	19	63.16	linear
Windows 2000	closed	358	9.35	linear
Windows XP	closed	297	8.97	linear
Mac OS X	closed	300	4.64	linear
Red Hat Enterprise 4	open	264	5.48	nicht linear
Debian 3.1	open	207	6.45	nicht linear
mySQL 5	open	33	46.00	linear
PostgreSQL 8	open	25	58.96	linear
Oracle 10g v8	closed	63	29.75	nicht linear
DB2 v8	closed	13	136.38	linear

In der ersten Untersuchung wird die Anzahl an Bugs (#Vuln), die mittlere Zeit in Tagen zwischen veröffentlichten Sicherheitslücken (MTBVD) und die Entwicklung der Anzahl an Schwachstellen über die Zeit (EASZ) für die verschiedenen Softwareprojekte verglichen (Tabelle 3). 12 von 17 Projekten folgen dabei einer linearen Entwicklung für die Anzahl der Fehler über die Zeit, was einer konstanten Rate mit der Fehler gefunden werden entspricht. Die Werte für die verschiedenen Projekt gehen zwar weit auseinander, beim Vergleich von Open-Source mit Closed-Source Software kann allerdings kein signifikanter Unterschied festgestellt werden. Auch zwischen Open-Source Projekten die nach dem „Kathedrale"-Modell und Projekten die nach dem „Basar"-Modell entwickelt wurden lassen sich keine Unterschiede feststellen.

Tabelle 4: [3] Schwere der Schwachstellen nach CVSS (2011)

Software	Mittelwert	Median	CVSS > 7
IE 7	6.65	6.80	45.95%
Firefox	6.38	6.40	36.53%
MS Outlook Express 6	6.18	5.10	39.13%
Thunderbird 1	6.53	6.80	47.27%
IIS 5	6.00	5.00	36.14%
Apache 2	5.36	5.00	18.75%
MS Office 2003	8.11	9.30	67.72%
OpenOffice 2	7.61	7.60	63.16%
Windows 2000	6.58	7.20	57.92%
Windows XP	6.67	7.20	57.92%
Mac OS X	6.18	6.80	41.33%
Red Hat Enterprise 4	4.72	4.90	23.11%
Debian 3.1	4.75	4.90	23.19%
mySQL 5	5.05	4.90	12.12%
PostgreSQL 8	6.17	6.80	36.00%
Oracle 10g v8	5.96	5.50	33.33%
DB2 v8	6.22	7.20	53.85%

In der zweiten Untersuchung werden die Softwareprojekte darauf verglichen, wie schwerwiegend die aufgetreten Sicherheitslücken sind. Dazu werden die Werte aus dem Common Vulnerability Scoring System (CVSS) verwendet, bei dem Sicherheitslücken von Sicherheitsexperten und Softwareanbietern im Konsens auf einer Skala von 0 bis 10 (für schwerste Fehler) eingestuft werden (Tabelle 4). Der Anteil der schweren Sicherheitslücken (CVSS Wert höher als 7) an der Gesamtanzahl, der durchschnittliche CVSS Wert und der Median der Einstufungen werden dazu bestimmt. Auch hier ergibt die statistische Analyse jedoch keine signifikanten Unterschiede zwischen Open-Source und Closed-Source Projekten.

Tabelle 5: [3] Gepatchte und ungepatchte Schwachstellen (2011)

Software	ungepatcht	Median CVSS ungepatcht	Median CVSS gepatcht
IE 7	66.22%	5.0	9.3
Firefox	20.36%	5.0	6.8
MS Outlook Express 6	65.22%	5.0	7.3
Thunderbird 1	5.45%	3.45	6.95
IIS 5	48.19%	5.0	7.2
Apache 2	26.25%	4.7	5.0
MS Office 2003	4.04%	5.05	9.3
OpenOffice 2	21.05%	5.25	9.3
Windows 2000	30.39%	5.1	9.3
Windows XP	30.64%	5.0	7.2
Mac OS X	6.67%	5.0	6.8
Red Hat Enterprise 4	14.77%	4.9	4.9
Debian 3.1	14.48%	4.9	4.9
mySQL 5	24.24%	4.6	4.9
PostgreSQL 8	12.00%	9.0	6.3
Oracle 10g v8	12.70%	7.35	5.5
DB2 v8	-	-	-

Mit den Daten aus der NVD der NIST und den Angaben auf den Entwicklerseiten zu den Veröffentlichungsdaten von Patches ist auch eine Untersuchung des Patchverhaltens der Entwickler möglich (Tabelle 5). Es ist zu beobachten, dass sich Microsoft auf das patchen von schweren Fehlern konzentriert und weniger schwere Fehler nicht patched, während die meisten anderen Hersteller wie zum Beispiel Apple und Open-Source Entwickler versuchen alle Fehler zu patchen. Man kann daraus schließen, dass das Patchverhalten stark von den Entwicklern und deren Prioritäten abhängt und nicht von der angewandten Entwicklungsmethode. Ein Unterschied der auf die verwendete Entwicklungsmethode zurückzuführen wäre ist auch hier nicht erkennbar.

Das Gesamtergebnis der Studie ist damit, dass nicht die Entwicklungsmethode für die Sicherheit entscheidend ist, sondern die jeweilige Strategie, wie mit den Softwarefehlern umzugehen ist. Um die Software-Sicherheit zu verbessern wäre deshalb ein möglicher Ansatz, stärkere ökonomische Anreize zu geben, Sicherheitslücken zu patchen. Das Ergebnis dieser Studie, dass Open-Source und Closed-Source Entwicklungsmethoden zur gleichen Software-Sicherheit führen ist allerdings nur auf große und weitverbreitete Softwareprojekte anzuwenden. Für einen Vergleich von kleinen Softwareprojekten ohne große öffentliche Aufmerksamkeit ist eine empirische Analyse dagegen nicht möglich, da dafür zu den jeweiligen Projekten zu wenig Daten vorliegen.

5.2 Einsatz von Open-Source Software in Firmen und Behörden

Open-Source Software hat sich zu einem Milliardenmarkt entwickelt. Die Softwarefirma Red Hat hatte im letzten Jahr einen Umsatz von 1,5 Milliarden US-Dollar [20], die Firma Mozilla etwa 300 Millionen US-Dollar [21]. Das Geschäftsmodell der Firmen die Open-Source Software entwickeln unterscheidet sich dabei von dem herkömmlicher Softwareentwickler. Umsatz wird hauptsächlich durch das Anbieten von Support und zusätzlichen Services zu der Software generiert. Die Firma Red Hat verkauft zwar ihre Linux-Versionen mit integriertem Support, stellt aber gleichzeitig den Source-Code, mit dem das Betriebssystem nachgebaut werden kann, kostenlos zur Verfügung. Mozilla macht den Hauptteil ihrer Einnahmen durch eine Partnerschaft

doi: 10.2313/NET-2014-08-1_20

mit Google, indem Google als Standard Suchmaschine im Browser voreingestellt ist.

Laut einer Umfrage des Marktforschungsinstitut Gartner [7] setzten im Jahr 2008 im Raum Europa, Nordamerika und Asien 85% der Unternehmen Open-Source Software ein. 69% der Firmen haben allerdings keine spezifischen Vorgaben für den Einsatz von Open-Source Software, die die rechtlichen Besonderheiten regeln, sondern setzen diese wie kommerzielle Software ein.

In Deutschland wurde eine Umfrage zu dem Thema Open-Source Einsatz in Unternehmen im Jahr 2009 von heise open und der Wilken GmbH durchgeführt [8]. Dabei gaben 40% der Unternehmen an, dass Open-Source von unternehmenskritischer Bedeutung ist, wobei Open-Source Software umso eher in unternehmenskritischen Bereichen eingesetzt wird, je länger ein Unternehmen Erfahrungen mit Open-Source Software gesammelt hat. Das bedeutet, dass der Einsatz in unternehmenskritschen Bereichen in Zukunft weiter ansteigen wird.

Weiterhin untersuchte die Studie die Gründe für den Einsatz von Open-Source Software, wobei Mehrfachnennungen möglich waren. Dabei war der am häufigsten angegebene Grund mit 90%, dass Open-Source Software vor allem eingesetzt wird, weil das Management Lizenzkosten sparen will und Open-Source als kostenlose Alternative sieht. Bei kleinen und neu gegründeten Unternehmen ist dieses Motiv besonders wichtig. An zweiter stelle steht mit 70% die Herstellerunabhängigkeit und etwa 60% geben an, dass sie Open-Source Software wegen der Software-Sicherheit verwenden. Nur etwa 40% gaben an, dass sie Open-Source Software wegen der Verfügbarkeit des Quellcodes einsetzen, und noch weniger mit 30% weil sie Fehler in der Software selber finden und beheben können. Open-Source kommt in den Firmen besonders häufig in Infrastrukturbereichen zum Einsatz, die Funktionalität über das Internet bereitstellen: Als Server-Betriebssystem (84%), Webserver (81%), Datenbank (79%) und in der Netzwerkinfrastruktur (73%). 57% der Firmen geben an, dass in mindestens einem Bereich Linux als Betriebssystem zum Einsatz kommt. Weltweit wird auf aktiven Webseiten zu 52% der Apache Webserver verwendet [9] (Stand März 2014).

Behörden müssen vor dem Einsatz von Software in sicherheitsrelevanten Bereichen den Quellcode der Software von Drittfirmen überprüfen lassen um die Funktionalität der Software zu verifizieren. Closed-Source Software für die der Code nicht von unabhängigen Dritten überprüft werden kann sind damit keine Alternative. Große Softwarehersteller haben darauf reagiert und Schnittstellen bereitgestellt über die gegen Entgeld der Quellcode zur verfügung gestellt wird. Microsoft hat dazu zum Beispiel die Shared-Source Initiative gegründet. Nach einer Studie der MITRE Corp. [10] sind bereits über 230 Open-Source Softwareprojekte in US Bundesbehörden im Einsatz. Deshalb hat des Department of Homeland Security das „Vulnerabilty Discovery and Redemption“ Programm gestartet, das täglich die Sicherheit von etwa 40 Open-Source Anwendungen, darunter Linux, Apache und MySQL, überprüft.

Open-Source Software kommt in allen Branchen hauptsächlich im Bereich der Netzwerkinfrastruktur zum Einsatz, was bedeutet, dass die Sicherheit der Firma bei Angriffen über das Internet direkt von der Sicherheit der verwendeten Open-Source Software abhängt. Da die Sicherheit von Software aber nicht im voraus bestimmt werden kann, werden bei der Entscheidung welche Software eingesetzt werden soll hauptsächlich die entstehenden Kosten betrachtet. Dieses Kriterium schließt vor allem auch den personellen Aufwand ein, der bei der Installation und Wartung entstehen.

6. HEARTBLEED

Als Heartbleed wird eine Schwachstelle in der Heartbeatfunktion des Transport Layer Security (TLS) Protokolls der OpenSSL Bibliothek bezeichnet, die in den Versionen 1.0.1 bis 1.0.1f aufgetreten ist. Da TLS keine Verbindung aufrecht halten kann ohne ständigen Datenaustausch, wird wenn gerade keine Nachrichten zwischen den beiden kommunizierenden PCs übertragen werden eine Heartbeatnachricht ausgetauscht. Dazu schickt ein PC einen Heartbeatrequest mit einem Payload, der aus einer zufällig generierten Nachricht mit einer zufälligen Länge besteht, an den anderen PC. Dieser schickt dann eine Kopie des erhalten Payloads in einer Heartbeatresponse zurück um die bestehende Verbindung zu bestätigen. Die Nachricht ist dabei folgendermaßen aufgebaut: Aus dem ersten Byte der Nachricht kann der Empfänger erkennen, dass es sich um den Heartbeat-Nachrichtentyp handelt. Die nächsten 2 Bytes enthalten die Länge des Heartbeat-Payloads und die restlichen Bytes sind der eigentliche Payload.

Beim erstellen des Buffers für die Heartbeatresponse und dem Kopieren der Payloadnachricht wird allerdings nicht überprüft, ob der Payload überhaupt so lang war wie die angegebene Länge in dem Request. Das heißt, dass wenn die Payloadnachricht kürzer war als die angegebene Länge, der Rest des Buffers für die Heartbeatresponse-Nachricht mit dem Inhalt im Speicher nach dem eigentlichen Payload gefüllt und als Antwort zurückgesendet wird. Die maximale Länge der Nachricht ist 64 Kilobyte groß. Da der Speicher der für den Payload der Heartbeatrequest angelegt wird in der OpenSSL Bibliothek liegt, können hier die privaten Schlüssel, die der PC für die asymmetrisch verschlüsselte Kommunikation benötigt liegen. Weitere Daten die ausgelesen werden könnten sind zum Beispiel Usernamen, Passwörter, private Nachrichten oder Emails.

Genau so einfach, wie der Fehler auszunutzen ist, konnte er auch in der OpenSSL Version 1.0.1g behoben werden. Dazu wird, wenn die angegebene Länge des Payloads größer ist als die tatsächliche Länge, der Heartbeatrequest verworfen und keine Heartbeatresponse zurückgeschickt.

6.1 Zeitlicher Ablauf

Am 15. Dezember 2011 schlägt Robin Seggelmann einen Patch vor, der das Heartbeatprotokoll an dessen Entwicklung er beteiligt war, zu OpenSSL hinzufügt [11]. Nach einigen Anpassungen wird der Patch am 31. Dezember 2011 von Stephen Henson der dem Entwicklerteam von OpenSSL angehört in den Hauptcodezweig aufgenommen. Dabei übersieht dieser die fehlende Längenabfrage für den Payload und fügt so den Heartbleed Bug in die offizielle OpenSSL Version ein.

doi: 10.2313/NET-2014-08-1_20

Der Fehler bleibt danach über zwei Jahre lang unentdeckt im Code bis ihn Neel Mehta, der für Google Security arbeitet, am 21. März 2014 findet. Den weiteren Verlauf im Umgang mit der Schwachstelle untersucht Ben Grubb in seinem Artikel „Heartbleed disclosure timeline: who knew what and when“ [12]. Noch am selben Tag wird der Fehler auf allen Servern von Google behoben. Am 1. April 2014 informiert Google das OpenSSL Entwicklerteam über die gefundene Schwachstelle, gibt aber keine weiteren Informationen darüber weiter, welche Partner von Google bereits über den Fehler informiert wurden.

Am 2. April findet die finnische IT-Sicherheitsfirma Codenomicon unabhängig von Google den selben Fehler, informiert daraufhin das „National Cyber Security Centre Finland“ und erfindet den später weitverbreiteten Namen Heartbleed. Einige weitere Firmen, inklusive Red Hat und Facebook, wurden von Google, Codenomicon oder der OpenSSL Entwicklergemeinde vor der Veröffentlichung des Fehlers informiert. Wie der Informationsaustausch genau ablief ist aber unbekannt, da jeder der informiert wurde ein Non-Disclosure-Agreement unterzeichnen musste. Am 7. April informiert Codenomicon ebenfalls das OpenSSL Entwicklerteam, das daraufhin entscheidet noch am selben Tag eine neue Version zu veröffentlichen, da sie das Risiko jetzt höher einschätzen wenn zwei Teams den Fehler unabhängig voneinander fast gleichzeitig gefunden haben. Codenomicon veröffentlicht daraufhin einen Tweet in dem sie erklären, dass sie den Bug ebenfalls gefunden haben und verlinken auf ihre Seite Heartbleed.com. Auf dieser Seite ist das von Codenomicon entworfene Logo für den Bug zu sehen und Erklärungen wie der Bug funktioniert. Innerhalb von 24 Stunden nach der Veröffentlichung wurden die ersten erfoglreichen Angriffe mit Heartbleed durchgeführt [13].

Das Content Delivery Network Cloudflare, das bereits vorab informiert wurde, veröffentlicht ebenfalls am 7. April einen Blogeintrag über die Hintergründe des Fehlers und wie man die alte OpenSSL Versionen so erstellen kann, dass der Fehler nicht auftritt. Noch am selben Tag erstellt Ubuntu einen Patch für das Betriebssystem. Am 9. April beschweren sich die Debian und Red Hat Entwickler darüber, dass der Bug unkoordiniert und trotz des verhängten Embargos veröffentlicht wurde und sie so mit einer Vorwarnzeit von nur 2 Tagen keine ausreichenden Vorbereitungen treffen konnten. Außerdem erhält Neel Mehta von Facebook und Microsoft über das Internet Bug Bounty Programm 15,000 US-Dollar, die er an die Freedom of the Press Foundation spendet.

Da CloudFlare nach eigenen Untersuchungen [14] zu dem Schluss gekommen ist, dass es so gut wie ausgeschlossen ist, dass mittels Heartbleed private Schlüssel des Servers ausgelesen werden können, starten sie am 11. April auf dem Firmenblog die Heartbleed Challange, die dazu aufforderte die privaten Schlüssel eines Servers durch Ausnutzung des Heartbleed Bugs herauszufinden und einzusenden. Dafür setzten sie einen nginx Server auf, auf dem eine OpenSSL Version lief, die mit dem Heartbleed Bug angreifbar ist. Bereits 9 Stunden später wurden die korrekten Schlüssel zwei mal eingesendet. Die beiden Gewinner, Softwareentwickler Fedor Indutny und Ilkka Mattila vom National Cyber Security Centre Finland, hatten dazu jeweils 2.5 Millionen bzw. 100,000 Heartbeatrequests an den Server gesendet. In dem Posting, das die beiden Gewinner bekannt gibt, vermutet der Autor Nick Sullivan [15], dass die Schlüssel ausgelesen werden konnten, weil der Server während der laufenden Challange neugestartet wurde. Aufgrund des Ergebnisses der Heartbleed Challange werden bei CloudFlare bis zum 17. April alle bestehenden SSL-Zertifikate durch neu ersetzt.

Am 14. April wird über den ersten Angriff durch Heartbleed berichtet. Die kanadische Steuerbehörde meldet, dass die Sozialversicherungsnummern von etwa 900 Steuerzahlern durch Ausnutzen der Heartbleed Schwachstelle entwendet wurden.

6.2 Grund für die große Aufmerksamkeit

Laut einer Umfrage des Pew Research Center [16] haben bis Ende April 60% der US-Bürger von Heartbleed gehört und 39% der Internetnutzer geben an aufgrund von Heartbleed ihre Passwörter geändert zu haben. Für eine Sicherheitslücke in einer Open-Source Bibliothek sind diese Bekanntheitswerte sehr hoch und eine Folge der breiten Berichterstattung darüber in den öffentlichen Medien. Durch die weitverbreitete Nutzung der OpenSSL Bibliothek auf Servern von IT-Firmen wie Amazon, Facebook, Google, Netflix und Yahoo ist praktisch jeder direkt durch die Sicherheitslücke betroffen gewesen. Insgesamt waren laut Dan Goodin (Ars Technica) [17] zwei drittel aller Webseiten im Internet betroffen. Laut Nicole Perlroth [18] (The New York Times) haben aber auch Wi-Fi Geräte von Cisco und etwa 50 Millionen Android Smartphones mit der Version 4.1.1, sowie Regirungsbehörden wie das F.B.I. und das Pentagon eine Fehlerhafte OpenSSL Version verwendet.

6.3 Ursachen für Heartbleed und Reaktionen

Der Fehler selbst ist leicht zu sehen, blieb aber über 2 Jahre lang unentdeckt Bestandteil der OpenSSL-Bibliothek. Das zeigt, dass obwohl die Bibliothek von vielen Firmen eingesetzt wurde niemand die betroffene Stelle im Quellcode überprüft hat, was der viele Augen Theorie des Open-Source Modells widerspricht. Für diesen Widerspruch gibt es verschieden Gründe die in diesem Fall zusammenfielen:

- Obwohl OpenSSL von vielen großen IT-Firmen eingesetzt wird blieb das Entwicklerteam im Kern das selbe und ist nicht mit gewachsen. Auf der Webseite des Projekts wird angegeben, dass nur ein Entwickler, Dr. Stephen N. Henson, Vollzeitprogrammierer ist und drei weitere freiwillige Entwickler zum Kernteam gehören. Insgesamt erhielt das Entwicklerteam etwa 2000 US-Doller an Spenden pro Jahr [18], was zeigt, dass bei den Firmen, die die OpenSSL Bibliothek verwendeten keine Zahlungen an die Entwickler der eingesetzten Open-Source Software eingeplant wurden.

- Die OpenSSL Bibliothek gilt als sehr sicher und durch die weite Verbreitung wurde angenommen, dass der Quellcode gut überprüft ist. Im Endeffekt haben aber durch diesen Effekt weniger Reviewer den Quellcode überprüft.

- Es gibt für Firmen keine Anreize die Sicherheit der verwendeten Open-Source Projekte selbst zu überprüfen

doi: 10.2313/NET-2014-08-1_20

und da sich Softwarefirmen heute sehr schnell anpassen und neue Dienste entwickeln müssen, wird in der Projektplanung der Aspekt der Software-Sicherheit vernachlässigt. Das hat dazu geführt, dass bis heute keine industrieweiten einheitlichen Sicherheitsstandards für die Entwicklung von Software durchgesetzt werden konnten.

- Der Code der OpenSSL Bibliothek wird zwar laufend von Codeanalyseprogrammen überprüft, laut David A. Wheeler [19] ist aber keines bekannt, das den Heartbleed Bug gefunden hätte.

Das OpenSSL Projekt besteht bereits seit 1998. Dadurch wurden immer mehr Erweiterungen der Bibliothek hinzugefügt, die von den meisten Anwendern nie benötigt werden. Jede dieser zusätzlichen Funktionalitäten erhöhen das Risiko, dass ein sicherheitsrelevanter Fehler in der Bibliothek auftritt. Aus diesem Grund entschieden sich die OpenBSD Entwickler das LibreSSL-Projekt zu starten, in dem als Grundlage der OpenSSL Quellcode der Version 1.0.1g verwendet wird, aber überflüssige Erweiterungen entfernt werden.

Als Reaktion auf das Bekanntwerden der Diskrepanz zwischen der finanziellen Unterstützung des Projekts und dem weitverbreiteten Einsatz der Bibliothek in kritischer Infrastruktur wurde von der Linux Foundation die Core Infrastructure Initiative gegründet. Mit ihr sollen Stellen in Open-Source Projekten finanziert werden, die mit Programmieren besetzt werden die sich ausschließlich mit Software-Sicherheit beschäftigen. Die Initiative wird finanziell unterstützt von 12 großen IT-Firmen, die jeweils 100.000 US-Dollar pro Jahr beisteuern, und hat damit in den nächsten 3 Jahren insgesamt 3,6 Millionen US-Dollar zur Verfügung. Die Initiative wird dem OpenSSL Projekt 2 Stellen für Vollzeitprogrammierer finanzieren.

7. ZUSAMMENFASSUNG

Die Debatte über die Sicherheit von Open-Source Software ist in der Öffentlichkeit angekommen. Durch ihren weitverbreiteten Einsatz in kritischer Infrastruktur, sind heute viele Bereiche des Alltags direkt davon betroffen. Während die Softwarelösungen immer komplexer werden, reichen die entwickelten Modelle zur Analyse der Einflüsse im Entwicklungsprozess auf die Sicherheit dieser Anwendungen noch nicht aus. Anhand empirischer Studien konnte aber gezeigt werden, dass die Wahl der Entwicklungsmethode, Open-Source oder Closed-Source, alleine keinen Einfluss auf die Software-Sicherheit hat. Vielmehr sind die Vorgaben für Sicherheitsstandards im Entwicklerteam entscheidend. Hier kann man ansetzen und einheitliche Standards in der Softwareentwicklung einführen, die von jeder Software, die in kritischen Bereichen eingesetzt werden soll befolgt werden müssen.

Bei der Suche nach den Ursachen des Heartbleed Bugs ist klar geworden, dass Firmen die Open-Source Software einsetzen dabei hauptsächlich auf die Kosten achten und von sich aus keinen Beitrag zur Verbesserung der Softwarequalität leisten. Sollen in Zukunft Fehler in diesem Ausmaß vermieden werden, müssen ökonomische Anreize für Firmen geschaffen werden sich gemeinsam mit den Entwicklern an der Umsetzung der ursprünglichen Idee hinter Open-Source zu beteiligen.

8. LITERATUR

[1] H. Härtig, C.-J. Hamann, M. Roitzsch: *The Mathematics of Obscurity: On the Trustworthiness of Open Source*, In Proceedings of the Ninth Workshop on the Economics of Information Security (WEIS), 2010

[2] G. Schryen, R. Kadura: *Open Source vs. Closed Source Software Towards Measuring Security*, In Proceedings of the 2009 ACM symposium on Applied Computing, pages 2016-2023, ACM New York, NY, USA, 2009

[3] G. Schryen: *Is Open Source Security a Myth?*, In Communications of the ACM, Volume 54 Issue 5, pages 130-140, ACM New York, NY, USA, 2011

[4] E. S. Raymond: *The Cathedral and the Bazaar*, O'Reilly & Associates, Inc. Sebastopol, CA, USA, 1999

[5] E. Levy *Wide Open Source*, `www.securityfocus.com/news/19`

[6] O. Alhazmi, Y. Malaiya, I. Ray: *Measuring, analyzing and predicting security vulnerabilities in software systems*, Computers & Security, Vol. 26, No. 3., pages 219-228, 2007

[7] *Gartner Says as Number of Business Processes Using Open-Source Software Increases, Companies Must Adopt and Enforce an OSS Policy*, `http://www.gartner.com/newsroom/id/801412`

[8] O. Diedrich *Trendstudie Open Source*, `http://www.heise.de/open/artikel/Trendstudie-Open-Source-221696.html`

[9] Netcraft *April 2014 Web Server Survey*, `http://news.netcraft.com/archives/2014/04/02/april-2014-web-server-survey.html`

[10] R. Narine *DHS backs open-source security*, eWeek, 2006

[11] *#2658: [PATCH] Add TLS/DTLS Heartbeats*, `http://rt.openssl.org/Ticket/Display.html?id=2658\&user=guest\&pass=guest`

[12] B. Grubb *Heartbleed disclosure timeline: who knew what and when*, `http://www.smh.com.au/it-pro/security-it/Heartbleed-disclosure-timeline-who-knew-what-and-when-20140415-zqurk.html`

[13] N. Perlroth *Heartbleed Internet Security Flaw Used in Attack*, `http://bits.blogs.nytimes.com/2014/04/18/Heartbleed-internet-security-flaw-used-in-attack/`

[14] N. Sullivan *Answering the Critical Question: Can You Get Private SSL Keys Using Heartbleed?*, `https://blog.cloudflare.com/answering-the-critical-question-can-you-get-private-ssl-keys-using-heartbleed`

[15] N. Sullivan *The Results of the CloudFlare Challenge*, `http://blog.cloudflare.com/the-results-of-the-cloudflare-challenge`

[16] PewResearch *Heartbleed's Impact*, `http://www.pewinternet.org/2014/04/30/heartbleeds-impact/2/`

[17] D. Goodin *Critical crypto bug in*

OpenSSL opens two-thirds of the Web to eavesdropping, http://arstechnica.com/security/2014/04/critical-crypto-bug-in-OpenSSL-opens-two-thirds-of-the-web-to-eavesdropping/

[18] N. Perlroth *Heartbleed Highlights a Contradiction in the Wen*, http://www.nytimes.com/2014/04/19/technology/heartbleed-highlights-a-contradiction-in-the-web.html

[19] D. A. Wheeler *How to Prevent the next Heartbleed*, http://www.dwheeler.com/essays/heartbleed.html

[20] Red Hat *Red Hat Reports Fourth Quarter and Fiscal Year 2014 Results*, http://www.redhat.com/about/news/press-archive/2014/3/red-hat-reports-fourth-quarter-and-fiscal-year-2014-results

[21] David Murphy *Google Paying Mozilla Almost $1B for Firefox Search: Why?*, http://www.pcmag.com/article2/0,2817,2398046,00.asp

doi: 10.2313/NET-2014-08-1_20

Abuse Detection

Peter Gawronski
Betreuer: Marcel von Maltitz
Seminar Innovative Internettechnologien und Mobilkommunikation SS2014
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: peter.gawronski@tum.de

KURZFASSUNG

Wenn Nutzerdaten in sozialen Netzwerken gesammelt existieren, entsteht ein Anreiz für Angreifer, teils aus kriminellen Kreisen, diese Daten auszulesen. Der Betreiber versucht dies durch geeignete Vorkehrungen zu verhindern, sodass ein Wechselspiel zwischen den zwei Kräften entsteht.
Zunächst werden mehrere Vorgehen, Typen und Ziele von Angriffen erläutert, wie Angreifer versuchen, in das Netzwerk einzudringen, um Daten aus ihm abzugreifen.
Im Bezug darauf werden drei verschiedene Methoden für das Aufspüren von unerwünschten Nutzern oder Verhaltens dargestellt, die aktuell von den Betreibern von sozialen Netzwerken verwendet werden (können) mit ihren Vor- und Nachteilen bei der Angriffsabwehr und Abwägungen zur Umsetzbarkeit. Konkrete Beispiele aus den zwei größten sozialen Netzwerken Facebook und Twitter zeigen den realen Einsatz solcher Mechanismen.

Schlüsselworte

Abuse Detection, Spam Detection, Phishing, Machine Learning

1. EINLEITUNG

Ziel beinahe aller Angriffe auf verschiedene soziale Netzwerke ist es, Nutzerinformationen aus diesen herauszuholen, um auf diesen Daten weiterzuarbeiten oder gleich als Rohdaten an Dritte weiterzugeben. Dahinter stehen häufig monetäre Interessen, da sich solch gewonnenen Daten auf Untergrundmärkten verkaufen lassen, dabei steigen die Preise für Datensätze mit detaillierter persönlicher Nutzerinformation. Doch auch neuere Angriffsmethoden müssen erkannt werden, wie Spear-Phishing, bei dem der User mit direkt auf ihn zugeschnittenen Botschaften kontaktiert wird und durch die personalisierte Form eher dazu bewegt wird, auf den Betrug hereinzufallen.

Daher ist es das Bestreben der Anbieter solcher Portale, unerwünschtes Verhalten einzudämmen und derartige Angriffe zu unterbinden. In der Realität wird eine schnelle Reaktion und zielgenaue Erkennung von Angreifern angestrebt, sodass der soziale Graph, also die Verknüpfungsstruktur zwischen den normalen Nutzern, nicht ausgespäht wird und weitgehend unangetastet bleibt.

Zunächst werden in Kapitel 2 verschiedene Angriffsziele und -strategien dargelegt, um dann in Abschnitt 3 auf die zugehörigen Abwehrmaßnahmen mit verschiedenen Konzepten einzugehen. Mögliche Reaktionen bei verdächtigen Nutzern oder eindeutigen Treffern werden in Kapitel 4 erläutert, reale Probleme der Defensivmaßnahmen in Kapitel 5.
Konkrete Umsetzungen solcher Defensivmaßnahmen finden sich in Kapitel 6 mit abschließender Zusammenfassung und Ausblick.

2. ANGREIFERSTRATEGIEN

2.1 Ziele des Angreifers/Typen von Abuse

Angreifer versuchen je nach Interessenslage andere Ziele zu erreichen und nutzen daher Angriffe vielfältiger Art, um erfolgreich zu sein. Dabei sind einige Typen miteinander verknüpft, so kann Spam Ausgangspunkt für weitere Angriffe sein, Phishing und Scam können massenweise wie Spam versendet werden. Dahinter stehen meist monetäre Interessen oder im nicht gewerblichen Bereich auch persönliche Schmähungen.

2.1.1 Spam

Spam ist die Verbreitung von unerwünschten Nachrichten wie Werbung, Kontaktgesuchen oder URLs zu unerwünschten Seiten. Diese Ausprägung des Angriffs existiert nun seit etwas mehr als 36 Jahren [26] und wächst beständig weiter. Das massenweise Auftreten kostet den Empfänger viel Zeit, und damit Geld, zum aussortieren und löschen, die niedrigen Eigenkosten des Senders durch die weltweite Vernetzung machen ihn fast omnipräsent. Daher ist es eines der Hauptziele von Abwehrmechanismen, Spam einzudämmen [16]

2.1.2 Scam

Scam ist wörtlich Vorschussbetrug, also das Versprechen von Leistungen oder Waren gegen Vorkasse, welche dann nicht eingehalten werden. Der Begriff wird häufig auch verwendet im Zusammenhang mit Schneeballsystemen, bei denen ein einfacher Geldverdienst angepriesen wird, dieser aber real nie eintritt [25]. Dies ist häufig eine Unterart von Spam, jedoch mit deutlich höherer krimineller Energie und steht im Zusammenhang mit vielerlei Betrugsfällen. Hier warnte sogar das BKA wegen solcher Fälle [21].

2.1.3 Phishing

Phishing ist der gezieltere Zugriff auf Zugangsdaten legitimer Nutzer, da diese bereits soziale Beziehungen im Netzwerk haben, die Angreifer damit unerkannt bleiben und ist eine Form des Identitätsdiebstahls.
Häufig ist hier ebenfalls Spam der Einstiegspunkt des Angriffs, dabei lässt sich durch das massenhafte versenden von

doi: 10.2313/NET-2014-08-1_21

Phishingnachrichten eine große Gruppe Nutzer erreichen, sodass der Phish trotz geringer Quote doch Erfolg hat.
Allgemeines Phishing kann gravierende Konsequenzen haben: Wenn ein legitimer Nutzer Opfer eines Phishings wird, ist er möglicherweise plötzlich Teil eines Botnetzwerks und gleichzeitig für die vorherigen Verfahren nur schwer ermittelbar.
Je nach abgegriffenen Logins können später weitere Zugangsdaten bezogen werden, z.B. bei der Übernahme eines Mailaccounts, auf den Passwort-Wiederherstellungsmails anderer Plattformen gesendet werden.
Aber auch direkter Schaden, wie Zugriff auf Bankkonten, Diebstahl sensitiver Daten oder Firmendokumenten etc. ist möglich.

„It doesn't matter how many firewalls, encryption software, certificates, or two-factor authentication mechanisms an organization has if the person behind the keyboard falls for a phish.“ (Aus Hong, J. [8])

Phishing ist bei gut vorbereiteten Attacken für den Angreifer relativ teuer, verspricht jedoch deutlich höheren Gewinn als die anderen Methoden.
E-Mails mit Aufforderungen von Fremden, Webseiten aufzurufen oder Software zu installieren, werden von erfahrenen Nutzern ignoriert oder zumindest kritisch hinterfragt. Ist die Mail jedoch von einem (scheinbaren) Freund oder übergeordneter Stelle wie dem Administrator der Firma, steigt die Glaubwürdigkeit und damit die Erfolgswahrscheinlichkeit [8].
Solch personalisierten Phishingmails, die möglicherweise an einen einzelnen Nutzer gehen, sind dann auch schwerer zu filtern, da sie normaler Korrespondenz sehr ähnlich sehen können. Ein Webseitenbesuch kann mittels vielfältiger Exploits zu Malware auf dem Rechner des Nutzers führen, wenn dieser nicht gut gesichert ist. Dies kann dann Schadensszenarien wie Datendiebstahl, Spionage, Passwort- oder Bankdatendiebstahl nach sich ziehen und dem Angreifer weiter nutzen.
Die Aufforderung zum Download von Software wird von den meisten Nutzern sehr kritisch betrachtet und ist auffälliger als das ausnutzen von Exploits, wird diese scheinbar legitime Software installiert ergeben sich die selben Konsequenzen wie vorher.

2.1.4 Social Graph Crawling

Die persönliche Verknüpfung einer Person innerhalb eines Netzwerkes hat einen hohen Wert, denn aus diesen Angaben können vielfältige Rückschlüsse auf den Nutzer gezogen werden. Solche Information kann für andere Angriffe genutzt werden wie z.B. personalisiertem Spam oder auf den Nutzer zugeschnittene Phishingversuche. Dabei kann der Angreifer einen Lawineneffekt nutzen, indem er von den Ergebnissen weitersucht und erhöht mit jeder Stufe des Angriffs den Umfang der abgegriffenen Daten um ein Vielfaches.

2.1.5 Motiv: Monetäre Interessen

Bei fast allen zuvor genannten Angriffszielen stehen Finanzen im Hintergrund. Aus der Investition in Angriffe soll ein größerer Gewinn abfallen, kriminell oder nicht, oder es soll zumindest einem anderen geschadet werden, ohne selbst Gewinn abzuschöpfen.
Hierbei zeigen sich zwei relativ neue Felder: Schaden für den Anbieter und/oder Gewinn für den Angreifer durch Werbung[20, 4].
Werbung ist im Internet weit verbreitet, daher wird dies ebenfalls ausgenutzt:
Ein Angreifer versucht mittels künstlich erzeugten Klicks seine eigene Seite in Suchmaschinen höher zu platzieren (z.B. über +1 von Google+) und schaltet dort Werbung. Durch die bessere Positionierung erreichen mehr Nutzer seine Seite und sehen die Werbung, also verdient der Angreifer Geld.
Dies ist häufig im Rahmen von sog. Suchmaschinenoptimierung anzutreffen, bei denen Firmen ähnliches ganz öffentlich anbieten, Suchmaschinenbetreiber sich jedoch dagegen sträuben.
Eine neuere Methode ist, dass ein Angreifer Bots Werbung auf Seiten sehen öässt, die der Werbende bezahlen muss und erzeugt so finanziellen Schaden.

2.1.6 Stalking, Bullying, Mobbing

Stalking, Bullying, Mobbing gehören neben anderem zu den privaten Fällen des Missbrauchs. Dieser ist meist persönlich motiviert und nicht gewerblich, wie die anderen Ausprägungen. Diese Form ist sehr spezifisch und meist auf einen engen Kreis begrenzt, teils nur bestehend aus Täter und Opfer.
Hier spielt die Form der psychischen Verletzung eine größere Rolle, monetäre Hintergedanken sind eher selten.

2.2 Vorgehensweisen

Für die Vielzahl an Angriffen gibt es eine entsprechende Anzahl an Möglichkeiten, den konkreten Angriff auszuführen, die es einem Angreifer ermöglichen, seine Ziele zu erreichen.

2.2.1 Fake Accounts

Geläufigste Version des Angriffs sind gefälschte Accounts mit frei erfundenen Daten, um sich damit im Netzwerk anzumelden und von da aus weiterzuarbeiten.
Auf dieser Basis können alle hier erwähnten Angriffe aufbauen, daher ist es für den Betreiber wichtig, solche Nutzer zu erkennen und zu entfernen.

2.2.2 Botnetze

Ein Botnetz ist ein Zusammenschluss von mehreren Computern, entweder vom Angreifer selbst gestellt oder durch Malware ferngesteuerte Fremdrechner, die dann Angriffe auf das Netzwerk starten. Angriffe auf soziale Netzwerke zielen dann unter anderem auf das Erstellen von Fake Accounts oder Accountübernahme.

2.2.3 Accountübernahme

Tritt häufig im Zusammenhang mit Phishing (vgl. [3, 8]) auf, dort wird nach erfolgreichem Phishing der Account infiltriert und Nutzen daraus gezogen. Dies wird teilweise auch durch Datendiebstahl vom Server realisiert, wenn Accountinformationen nicht gesichert gespeichert wurden.
Wie in Abschnitt 2.1.3 beschrieben, ist dieser Angriff sehr gefährlich für das Opfer, da der Angreifer uneingeschränkten Zugriff auf dessen Daten und Zugänge hat und diese für weitere Angriffe und Manipulation nutzen kann.

doi: 10.2313/NET-2014-08-1_21

3. ABWEHRSTRATEGIEN

3.1 Probleme, Beweggründe/Motive

Ein Anbieter sieht sich vor dem Problem, dass er Aufwand betreiben muss und damit Kosten hat, um Missbrauch innerhalb des Netzwerkes zu unterbinden. Mangelhafte Überwachung durch den Betreiber hat vielerlei Folgen:

3.1.1 Probleme

– Mehr ungewollte Werbung/Spam [6]:
Nutzer wollen Spam nicht sehen und werden unzufrieden, wenn sie zu viel erreicht. Großangelegte Kampagnen verschwenden überdies hinaus auch die Ressourcen des Anbieters und verursachen so Kosten.
– Falsch adressierte Werbung [nach [4]]:
Werbung an unerkannte Bots kostet den Werbetreibenden, dieser übt daher Druck auf den Betreiber aus, um auch nur für Werbung an reale Nutzer zu zahlen.
– Abgreifen des sozialen Graphen [1]:
Ein Angreifer versucht, die sozialen Kontakte eines Nutzers auszulesen. Je nach internen Richtlinien will der dies Betreiber verhindern oder zumindest soweit einschränken, dass es für einen Angreifer unattraktiv wird. Diese Information haben meist einen hohen Wert für Dritte, aber auch den Betreiber.

3.1.2 Konsequenzen

– Verfälschen von Ergebnissen [20, 14]:
Botnetze und Spammer können Sucheinträge verfälschen, so konnten bei Google Ergebnisse der Suche hervorgehoben werden durch Missbrauch der +1-Funktion oder bei Twitter unerwünschte Trendbegriffe entstehen. Derartigen Missbrauch von regulären Funktionen versucht der Betreiber zu vereiteln.
– Allgemein Schäden für den Betreiber, Nutzer oder legitimen Verwerter (z.B. legal Werbende):
Es gibt viele Arten des Missbrauchs, schlussendlich hat eine der Gruppen einen Schaden vielfältiger Art. Schäden für Nutzer oder Verwerter senken dabei die Zufriedenheit mit dem Anbieter, sie könnten abwandern. Daher ist es für den Betreiber erstrebenswert, dies effektiv zu verhindern.

Es ergibt sich eine weitere Notwendigkeit speziell bei persönlichen Angriffen aus Nutzer:
Die Plattform muss eine Möglichkeit besitzen, z.B. Mobbing oder Stalking oder Propaganda zu unterbinden. Dies ist mit den hier vorgestellten Methoden aber nur schwer möglich, da die Angreifer eine völlig andere Personengruppe darstellen und hier eher nur manuelle Überprüfung bleibt.

3.2 Abwehrtechniken

Drei verschiedene Techniken zur Erkennung bzw. Hervorhebung von Auffälligkeiten werden nachfolgend beschrieben und auf ihre Leistungsfähigkeit hin untersucht. Im Kapitel 6 werden hierzu real einsetzbare/eingesetzte Software gezeigt, die auf diesen Grundsteinen aufbaut.

Grundsätzlich ist die Abwehr ein Kreislauf, bei dem die Angreifer durch Verbesserung ihrer Angriffstechniken versuchen, mehr Daten abzugreifen, Spam zu verbreiten oder länger unerkannt zu bleiben und der Verteidigung, die diese Angriffsflächen abschirmt und schützt (vgl. Abb 1).

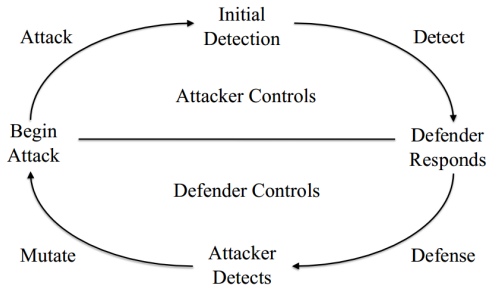


Abbildung 1: Der konfliktäre Zyklus, aus [10], S.1

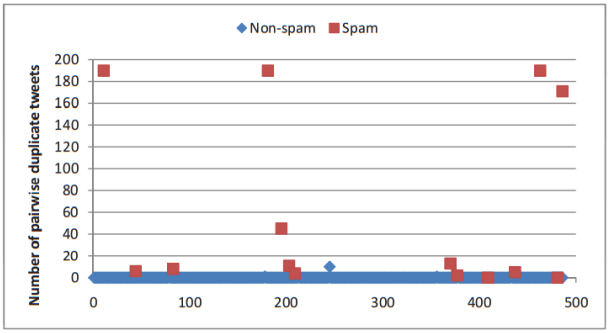
3.2.1 Machine Based Learning

Ein häufiger Vertreter der Abwehr von unerwünschten Vorkommnissen ist das Maschinelle Lernen (ML) [10, 13]. Hierbei ist die Idee, dass man aus Erfahrungswerten lernen kann, welches Verhalten wahrscheinlich einem normalen Nutzer zuzuordnen ist und welches auf eine Maschine oder einen bösartigen Nutzer deutet. Im einfachsten Fall ist dies lediglich ein Schwellwert, meist jedoch mehrere gemessene Eigenschaften, die zusammen eine Wahrscheinlichkeit ergeben, mit der es sich um einen legitimen Nutzer handelt.
Dabei werden häufig Daten und Werte aus den jeweiligen sozialen Strukturen ermittelt, aber auch welche aus dem Verhalten der Nutzer.
So sind geläufige Metriken für Verknüpfungen die Anzahl der Freunde/Follower, meist unterschieden nach gestellten Anfragen/Abonnements und erhaltenen Anfragen/Followern, zusätzlich der Quotient aus den zwei Werten.
Hierbei zeigt sich, dass normale Nutzer häufig andere Werte erreichen als künstliche Accounts, so stellen Roboter bspw. viele Freundschaftsanfragen, erhalten selbst aber eher kaum welche oder sind nur innerhalb eines eigenen Netzes aus Bots verbunden.

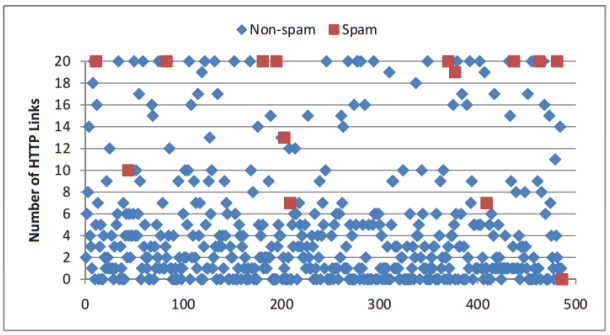
Weitere Kriterien können aus dem Verhalten, insbesondere aus dem Inhalt ihrer Postings, gezogen werden (Abb. 2 für Twitter):
– Die Anzahl von duplizierten Posts (Abb. 2 (a)) ist ein sehr starker Indikator für Spam, da ein normaler Nutzer eher nie einen gleichlautenden Post verschickt. Ein spammender Account wird, bis auf in der Abbildung gefilterte Veränderungen im Hashtag oder @-Bezeichner, deutlich eher solch duplizierte Posts versenden.
– Zusätzlich zu diesem starken Indikator gibt es schwächere Indikatoren wie die Anzahl der Links und Hashtags in den Posts (c), hier gibt es jedoch auch normale Nutzer, die ähnliche Werte erreichen.
– Verbessert werden können diese Metriken, wenn man übergreifend nach Spamwörtern aus aktuellen Kampagnen sucht (Abb. 8): Es zeigen sich deutliche Unterschiede in der Häufigkeit von genutzten Spamwörtern, sodass sich Spamaccounts leichter filtern lassen.

Das Gesamtkonzept des Maschinellen Lernens beruht darauf, gute Metriken zur Klassifizierung zu finden und regelmäßig anzupassen, da sich mit ihnen die Qualität des Ansatzes entscheidet.

doi: 10.2313/NET-2014-08-1_21



(a) The Number of Pairwise Duplications



(c) The Number of Links

Abbildung 2: Metriken zur Erkennung von Spammern bei Twitter, aus [14], S.8

3.2.2 *Social Graph Properties*

Betrachten der Eigenschaften des sozialen Graphen oder Verknüpfungsgraphen zwischen den Nutzern stellen einen unabhängigen Ansatz zum Maschinellen Lernen dar, denn hier wird weder auf das Verhalten, noch die direkte Beziehung unter den Nutzern geachtet, sondern der Gesamtgraph der Verknüpfungen und Beziehungen der Nutzer untereinander.

Begründet wird diese Vorgehensweise mit der hohen Wahrscheinlichkeit, dass Fake Accounts nur lose über wenige Beziehungen oder gar nicht mit realen Nutzern verknüpft sind, vielmehr häufig nur unter sich [4]. Reale Nutzer hingegen sind meist stark untereinander vermascht [4]. Damit ergibt sich eine Möglichkeit aus der Graphentheorie, die eher isolierten Gruppen des sozialen Graphen zu finden und diese als wahrscheinliche Bots zu markieren.

Zur Analyse werden bei Cao [4] mehrere Startpunkte gewählt, von denen dann das Netz abgelaufen/besucht wird. Dabei können verschiedene Metriken zu einzelnen Knoten im Netzwerk bestimmt werden.
So können isoliertere Gruppen im Netzwerk in den einzelnen Verfahren unterschiedlich schnell und präzise erkannt werden, sodass hier ebenfalls die Wahl eines geeigneten Algorithmus der Schlüssel zum Erfolg bleibt.

3.2.3 *Honeypots*

Ein bereits aus Zeiten vor sozialen Netzwerken stammendes Verfahren sind Honeypots, also in diesem Fall seinerseits künstliche Accounts, die jedoch von der Plattform betrieben werden und selbst keine Interaktion anstoßen. Daher ist es ungewöhnlich und verdächtig, wenn ein anderer Account mit diesem Fake Account kommuniziert oder Freundschaftsanfragen stellt/folgt/erwähnt [11].

Honeypots sind deshalb mächtig, weil ihnen jegliche Form von Abuse auffallen kann und sie unabhängig von Metriken und ähnlichen arbeiten können. Stringhini et al. [11] untersuchten dabei u.a. Facebook und Twitter mit einer großen Anzahl Honeypots und erhielten so Daten über Spamkampagnen und typisches Spamverhalten.

Durch Honeypots wird auch ersichtlich, welche Arten und wie viel Spam derzeit gesendet wird. Aus diesen Daten können dann z.B. Kriterien/Suchbegriffe zur weiteren Analyse für ML-Systeme generiert werden oder auch direkt Nachrichten an Kontrollpersonen des Netzwerkes zur manuellen Überprüfung von Accounts gesendet werden.

Nachteilig ist hier, dass Honeypots nicht aktiv suchen, sondern passiv warten und im Normalfall keine Verknüpfungen ins Netzwerk haben. Dies kann von intelligenten Angreifern erkannt und betreffende Accounts ignoriert werden, wenn sie merken, dass es sich wahrscheinlich um Honeypots handelt.

3.2.4 *Flagging*

Beim Flagging werden Nutzer oder Beiträge nicht automatisch, sondern durch einzelne Nutzer manuell markiert. An der Effizienz in Treffern je Meldung gemessen ist es ein schlechtes Verfahren, in der Literatur werden Quoten von ca. 5% genannt [4]. Jedoch ist dies das einzige Verfahren, dass gezielte Angriffe auf einzelne Nutzer wie Stalking, Mobbing und ähnliches abdecken kann.
Derartige Systeme sind auch außerhalb sozialer Netzwerke fast immer vorhanden und ermöglichen Nutzern, etwas direkt beim Betreiber anzuzeigen, sodass dieser geeignete weitere Schritte vornehmen kann.
Dieses Verfahren sei hier jedoch als Sonderfall zu betrachten, da es nicht automatisiert erfolgt.

3.3 Abwehr von Phishing

Vorhergehende Abwehrmaßnahmen sind hauptsächlich auf das Eindämmen von Spam ausgelegt und versuchen, falsche Accounts zu enttarnen, den Haupteinstiegspunkt für Spam. Phishing kann, wenn es wie Spam verteilt auf die Nutzer einwirkt, ähnlich wie dieser mittels automatisierten Einstufens durch Maschinen gefiltert werden.

Es gibt jedoch Möglichkeiten, auch stark personalisiertes Phising einzuschränken und aufzudecken (nach [8]), wenn der erste Schritt der Mailfilterung nicht erfolgreich war:
– Sichere Browser: Wie in Abb. 3 zu sehen, unterstützen viele Browser Blacklisten von phishing-verdächtigen Seiten, z.B. nutzt Mozilla Firefox eine von Google verwaltete Liste [23, 24]. Der Browser weist den Nutzer aktiv darauf hin, dass die derzeitige Seite wahrscheinlich nicht die ist, für die er sie hält. Bei besonderer Schwere blockieren Browser den Zugriff zunächst sogar komplett, um Exploits auf der Webseite zu verhindern. Auch Antivirensoftware schlägt bei solchen Bedrohungen häufig Alarm.
Zu sicheren Browsern gehört auch, dass sie sichere Verbindungen deutlich kennzeichnen wie in Abb. 4: Die originale

doi: 10.2313/NET-2014-08-1_21

Webseite wird zusätzlich als überprüft und sicher markiert.
– Sensibilisierung der Nutzer: Bereits häufig ist in E-Mails der Betreiber zu lesen, dass ihre Webseite einen Nutzer nie per Mail nach ihren Accountinformationen fragen würden, aber auch auf Webseiten selbst finden sich dazu Hinweise, die häufig für Phishingversuche nachgebaut werden.
Dieser Punkt ist für den Angreifer die größte Hürde: Ein aufmerksamer Nutzer erkennt Phishing meist leicht und selbst auf ausgeklügelten Seiten wird der Nutzer eher darüber nachdenken, geheime Informationen einfach preiszugeben oder lieber nochmals zu verifizieren, ob der Grund und die Seite legitim sind.

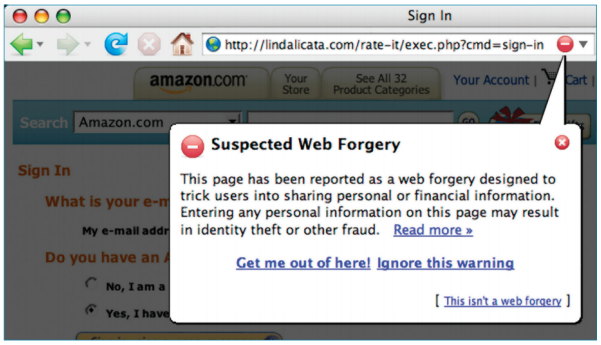


Abbildung 3: Aktive Warnung bei Firefox, aus [8], S.6

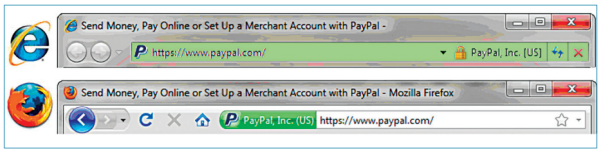


Abbildung 4: Validierung bei Internet Explorer und Firefox, aus [8], S.6

4. BEHANDLUNGSSTRATEGIEN

4.1 Fake Accounts

Wurde mittels eines der Verfahren aus Kapitel 3 mit hoher Wahrscheinlichkeit festgestellt, dass ein Account nicht einem echten Nutzer entspricht, sind mehrere Optionen möglich, diesen Account zu überprüfen:

4.1.1 Captchas

Captchas sind kurze Überprüfungswerkzeuge, die eine einfache Unterscheidung zwischen Mensch und Maschine ermöglichen sollen und sind weit verbreitet. Meist besteht ein Captcha aus einem kleinen Bild mit Textinhalt, der verzerrt/-verschleiert ist, sodass er für einen Menschen lesbar bleibt, eine automatische Texterkennung (sog. OCR: Optical Character Recognition) jedoch fehlschlägt. Dies soll automatisierte Anfragen an ein System verhindern, da für jede Anfrage im optimalen Fall nur ein Mensch die Aufgabe lösen kann, so die Kosten eines Angriffs steigen und damit vielleicht nicht mehr im akzeptablen/wirtschaftlichen Bereich liegen.

Bekanntestes Beispiel ist das 2009 von Google aufgekaufte reCAPTCHA, welchen dem Nutzer zwei Wörter oder Zahlen vorlegt und dieser diese in ein Textfeld tippt. Dabei ist lediglich ein Wort für die Erkennung relevant, das andere stammt aus nicht erkannten Worten aus der Digitalisierung von Büchern oder Straßenschildern von Google Streetview [18].
Vorteil ist die im Normalfall relativ einfache und kurze Überprüfungszeit für einen legitimen Nutzer, wohingegen ein Bot o.ä. scheitern wird und entfernt werden kann. Nachteilig ist die teils eingeschränkte Barrierefreiheit: Angreifer entziffern durch immer bessere OCR-Leser selbst schwierige Captchas, daher müssen diese wiederum komplexer werden. Schließlich werden sie aber auch für Menschen kaum lesbar, insbesondere bei eingeschränkter Sehfähigkeit.
Zu schwere Captchas erzeugen Frust bei legitimen Nutzern, daher ist die Anforderung an das System zweiteilig: Captchas sollen für Menschen einfach bleiben und nur Bots fernhalten und eine geringe Falschpositiv-Rate in Kapitel 3 gewünscht wird. Ein normaler Nutzer sollte im Normalfall nur bei der Registrierung ein einzelnes Captcha lösen müssen, um sich als Mensch zu identifizieren.

4.1.2 Überprüfung durch Menschen

Dies ist einer der am häufigsten eingesetzte Weg, Nutzer zu überprüfen und häufig auch die letzte bzw. endgültige Station bei der Bewertung, ob ein Account, Beitrag oder Verhalten um legitimen oder nicht legitim ist und welche Schritte eingeleitet werden müssen wie z.B. das Löschen des betreffenden Accounts oder Sperren/Verwarnungen gegen Nutzer, möglicherweise auch rechtliche Mittel.

Das spanische soziale Netzwerk Tuenti nutzte nach Cao [4] allein vierzehn Vollzeitbeschäftigte, um gegen Missbrauch vorzugehen. Facebook hat auch eine große Zahl Mitarbeiter, die sich ausschließlich mit der Bekämpfung von unerwünschtem Verhalten beschäftigen [10]. Abwehrstrategien haben auch immer das Ziel, diese Arbeitsleistung zu reduzieren bzw. die Trefferrate zu erhöhen, um die Effektivität des Aufwands zu steigern.

Vollständige Automatisierung ohne menschliche Hilfe lässt sich kaum erreichen, da es neben den Hauptangreifern wie Spammern oder Datensammlern auch die in Kapitel 2 genannten persönlichen Angriffe wie Mobbing oder Stalking gibt, welche durch Menschen von Hand gemeldet und von Menschen geprüft werden müssen. Auch anstößige Bilder müssen manuell von Menschen daraufhin geprüft werden, ob sie gegen die Nutzungsrichtlinien verstoßen und werden dann gelöscht.

4.1.3 Überprüfung befreundeter Accounts

Wie in Abschnitt 3.2.2 erläutert, sind Fake Accounts häufig gut untereinander vernetzt, haben jedoch nur sehr wenige Verbindungskanten zum Hauptgeflecht der Nutzer.
Daher erscheint es sinnvoll, ebenfalls die verknüpften bzw. befreundeten Nutzer zu überprüfen, ob sie nicht auch selbst nur künstliche Accounts sind, wenn sie nicht durch die Mechanismen sowieso schon markiert wurden [4].

4.2 Phishing

Wurde Phishing erkannt ist das Beseitigen der zugehörigen Webseite aus dem Netz von hoher Priorität, da ihr so nicht mehr Menschen zum Opfer fallen können [8]. Zwischenzeitlich sollte die Webseite von Browsern wie in 3.3 markiert

doi: 10.2313/NET-2014-08-1_21

und geblockt werden.
Ist ein Phishingversuch erfolgreich gewesen, sollte im Rahmen der Schadensbegrenzung der zugehörige Account gründlich untersucht und davon abhängige Dienste und Registrierungen ebenfalls überprüft werden und die zugehörigen Zugangsdaten geändert werden.

5. PROBLEME UND BESCHRÄNKUNGEN

Jeder der Ansätze aus Abschnitt 3 hat seine Beschränkungen und auftretenden Probleme, die für alle in ähnlichem Umfang gelten:

5.1 False Positives

Falsch positive Treffer sind legitime Nutzer, deren Verhalten dem eines Roboters zu sehr ähneln und daher auf der Liste der Verdächtigen landen. Soll nur ein Captcha gelöst werden, sind die meisten Nutzer noch bereit, dieses zu lösen. Wird der Aufwand größer, kann es jedoch schnell zu Ärger und Frust kommen, der sich dann negativ auf den Betreiber auswirkt. Dies passierte 2011, als Google seine internen Suchkriterien verschärfte [22] und zu viele falsch positive Treffer auftraten.
Daher müssen beim Entwurf und Test der Metriken beide Fehlerarten in einer Optimierungsphase berücksichtigt werden, damit weder zu viele Bots durchs Raster fallen, aber auch nicht viele Nutzer gestört werden.

5.2 Änderung des Botverhaltens

Dies ist ein direkter Teil des Zyklus aus Abbildung 1, in dem der Angreifer seine Methoden stetig nachbessert, um an den Abschirmmaßnahmen vorbeizukommen. Auch können atypische Bots für Spezialaufgaben genutzt werden (insbesondere personalisiertes Phishing), die dann von einer bestimmten Metrik schlicht nicht erkannt werden. Daher ist es für die Betreiber notwendig, mehrfache Absicherungen zu haben, sodass die Erkennung auch das neue Verhalten erkennt..

6. KONKRETE BEISPIELE

6.1 Facebook Immune System

Das FIS ist das Verfahren, welches von den Betreibern der Plattform selbst angewandt wird [10]. Dabei wird ein Ansatz des Maschinellen Lernens verwendet, der sich auf schnell ändernde Bedingungen anpassen kann und dabei quasi in Echtzeit auf das Geschehen reagiert, was bei einer Größe wie Facebook täglich ca. 25 Milliarden nutzergenerierten Aktionen entspricht.
Der Versuch des Auslesens des sozialen Graphen ist für die Entwickler ein Sonderfall, da der Angreifer, der ein Angriffsmuster erstellt, dieses möglichst lange unentdeckt lassen will, um weiter in den Graphen eindringen zu können.
Aufgebaut ist die FIS in verschiedene Teilbereiche, die ineinandergreifen: Klassifikatoren überwachen den Netzwerkverkehr und interagieren eng mit einem Regelwerk und auf der anderen Seite mit einer speziellen formalen Sprache zur Merkmalsfindung. Zusammengefasst entstehen so Regeln und Eigenschaften für die Kategorisierung von Verhalten und Beiträgen. Diese werden über Rückkopplungsschleifen zurückgespeist und rufen schlussendlich eine Reaktion, wie z.B. geeignete Maßnahmen bei Verdacht auf Fake Accounts.
Eines der Hauptaugenmerke der FIS ist das schnelle Anpassen der Umgebung auf eingehende Daten, um den Graph gegen alle Arten von Angriffen abzudichten und sich nicht auf ein spezielles Detail festzulegen. Dabei werden die häufiger auftretenden falsch positiven Treffer in Kauf genommen, denn eine leicht höhere Fehlerrate bei sehr wenig Nutzern trifft effektiv weniger Nutzer als eine niedrige Rate bei einer großen Masse an Betroffenen [10].

6.2 Automatisierter Angriff auf soziale Netzwerke

Balduzzi et al. [1] beschreiben einen einfachen, aber sehr effektiven Angriff auf verschiedene, große soziale Netzwerke: Diese erlauben es dem Nutzer, mittels der E-Mailadresse nach Freunden zu suchen und zeigen dann an, ob ein passender Account existiert.
Dieses Verhalten ist für den normalen Nutzer von Vorteil, da er so schnell viele Freunde und Bekannte erreicht, kann jedoch auch verschieden missbraucht werden:
- Gestohlene oder erfundene Mailadressen können verifiziert werden, ob eine reale Person dahintersteckt: Spamattacken treffen nun eher echte Nutzer.
- Werden auf verschiedenen Portalen die gleichen Adressen genutzt, handelt es sich wahrscheinlich um dieselben Nutzer.

Viel gravierender ist die lose Privatsphäreneinstellung vieler Portale: Die meisten zeigen öffentlich ein Foto, Freunde (und damit einen Teil des sozialen Graphen), Homepage, Geburtsdatum, Hobbies etc. Diese Information kann dann für gezielte Werbung oder auch für Phishingattacken ausgenutzt werden, die stark personalisiert durch die erhaltenen Informationen.
Auch der Weiterverkauf solcher Daten kann für den Angreifer lukrativ sein.
In dem Artikel zeigt sich ein Unbewusstsein der Plattformbetreiber gegen derartige Angriffe: Ihre Anfrage-APIs reagierten ohne Einschränkungen und sehr schnell auf E-Mail Abfragen, beispielsweise Facebook mit bis zu zehn Millionen Datensätzen am Tag von einem einzelnen Rechner aus.

Balduzzi et al. [1] zeigen auch Gegenmaßnahmen auf, wie die Nutzung von Captchas, die Beschränkung der Anzahl an Abfragen eines einzelnen Nutzers, eine Anfrageraten-Limitierung (z.B. wenige pro Woche) oder einer Verschleierung der Verknüpfung der Mailadresse zu Account.

6.3 SybilRank mit Random Walk

Ein von Cao et al. [4] entwickeltes Verfahren SybilRank nutzt zur Analyse des Netzes zufällige Einstiegspunkte in den (ungerichteten) sozialen Graphen und führt dann einen gekürzten Random Walk auf diesem aus, baut dabei ein Netz des Vertrauens auf und errechnet danach, wie eng ein Nutzer an diesem Netz hängt. Wie in Abb. 5 ersichtlich, haben echte Nutzer fast immer deutlich kürzere Wege, damit lassen sich Bots gut identifizieren.

SybilRank soll dabei laut Cao [4] deutlich effizienter sein als vergleichbare, ähnliche Verfahren und hat darüber hinaus in der Testumgebung eines spanischen sozialen Netzwerkes (Tuenti) sehr hohe Trefferquoten erzielt: 100% aus 50.000 sicheren Treffern und 90% aus 200.000 wahrscheinlichen Treffern, was über den Werten vergleichbarer ML-Techniken liegt und zeitlich effizienter ist als die verglichenen ähnlichen Ansätze.

doi: 10.2313/NET-2014-08-1_21

Da die Annahme lautet, dass Bots nur schlecht zu realen Nutzern vernetzt sind, fallen prinzipbedingt diejenigen Bots durchs Raster, die bereits länger operieren und daher relativ gut immersiert sind. Dies bemängelt Yang et al. [15] und stellt seinerseits ein ML-Ansatz mit neuen Metriken vor, die auch solche Nutzer noch erkennen kann.

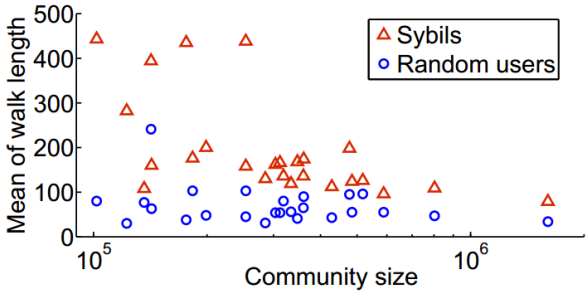


Abbildung 5: Mittlere Länge der Random Walks bei normalen Nutzern und Bots, aus [4] S.12

6.4 Maschinelles Lernen mit neuen Metriken

Yang et al. [15] stellen neue Metriken vor, die bereits existierenden deutlich überlegen sein sollen und sogar bessere Erkennung bieten als die zuvor genannten Eigenschaften des sozialen Graphen, Beispiele in Abb. 6 und 7. Beide Abbildungen zeigen die Verteilung (CDF) der jeweiligen Metriken, bezogen auf die Anzahl der versendeten Einladungen je Zeit, angenommener ausgehender und eingehender Freundschaftsanfragen und Verknüpfungsgrad des Nutzers. Auffällig sind die deutlichen Unterschiede der Anzahl angenommener ausgehender und eingehender Freundschaftsanfragen, dazu der Faktor der Verknüpfung der Freunde untereinander. Solche Differenzen eignen sich sehr gut für Klassifizierung mittels maschinellen Lernens, sodass laut Yang werden so auch bereits eingenistete Bots erkannt, die möglicherweise länger im Netzwerk sind und daher genügend Freunde haben, um nicht für Außenstehende gehalten zu werden.

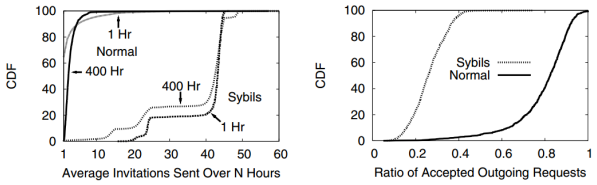


Abbildung 6: Vergleich zwischen Nutzer und Bot, aus [15], S.2

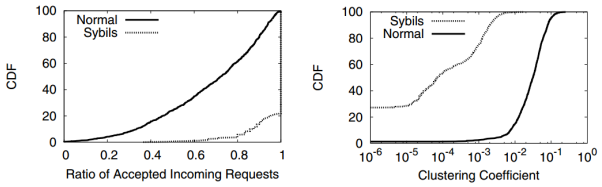


Abbildung 7: Vergleich zwischen Nutzer und Bots, aus [15], S.3

6.5 Honeypots: Analyse von Spam

Stringhini et al. [11] setzten eine Vielzahl von Honeypots in sozialen Netzwerken ein und stellten ihre Ergebnisse und Häufigkeiten der Spamkampagnen graphisch dar (Abb. 8). Dabei stellt der Durchmesser der Kreise das Volumen dar, aufgeteilt nach Kampagnentyp über die Zeit. Deutlich sichtbar ist die Verschiedenheit der einzelnen Kampagnen: So gibt es mehrere, die kontinuierlich laufen(1,5,6,7), zeitlich begrenzt sind (2,3,4,8) und teils sehr große Intensitäten erreichen können (4,6). Honeypots sind hier also sinnvoll, um das Ausmaß aktueller Angriffe auf verschiedenen Bereiche des Netzwerks zu erhalten, aber auch um z.B. die Effektivität von Abwehrmaßnahmen zu veranschaulichen.

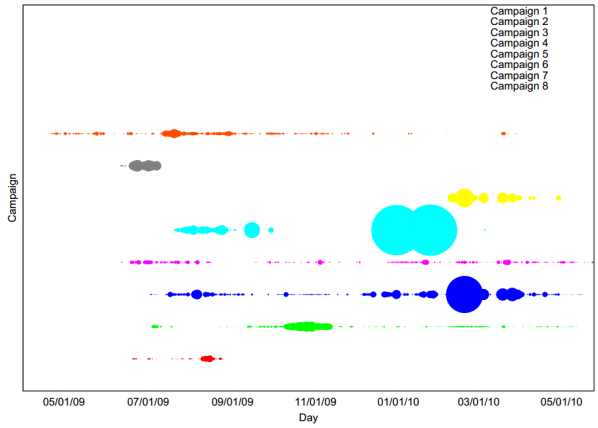


Abbildung 8: Spamkampagnen über Zeit, aus [11], S.8

7. ZUSAMMENFASSUNG/AUSBLICK

Insgesamt zeigt sich hier das Wechselspiel zwischen dem Angreifer und dem betroffenen Inhaber, der sein Netzwerk verteidigen muss. Die verschiedenen Beweggründe auf beiden Seiten werden den Wettlauf auch in Zukunft immer weiter treiben mit besseren Eindringstrategien auf der einen Seite und besseren Aufdeckmechanismen auf der anderen. Die Verteidigerseite versucht dabei den Großteil der Zeit die Oberhand zu behalten, da diese die Nutzerdaten schützt (vgl. Abb. 1) [10].

Maschinelles Lernen ist ein bewährtes Konzept, welches bereits länger aktiv eingesetzt wird und es daher viele Erfahrungswerte und Tauglichkeitsnachweise gibt. Es ist ein klassisches Verfahren, welches aus Eigenschaften eines Datensatzes Rückschlüsse auf die Eigenschaften zieht. Je nach Eignung und Auswahl der Parameter werden unterschiedliche Ergebnisse bei der Entdeckung gemacht, sodass diese Wahl essenziell ist und bei falscher Einstellung die Leistungsfähigkeit zunichte machen kann.

Hierbei sind Entwicklungen, wie die des Facebook Immune System [10] leistungsstarke Konzepte, wie aus einer großen Flut an eingehender Daten unerwünschte Nutzer oder Verhalten herausgefiltert werden können, um weitere Maßnahmen zu treffen. Das FIS nutzt dabei verhaltens- und verknüpfungsbasierte Parameter, um auffällige Nutzer zu zeigen.

doi: 10.2313/NET-2014-08-1_21

Echt graphbasierte Verfahren wie SybilRank [4] nutzen keine inhaltsbasierten Daten sondern beschränken sich auf das Auffinden von isolierten Gruppen innerhalb von Netzwerken, um künstliche Accounts zu identifizieren. Der soziale Graph als Ganzes wird hier statt eines Features des Accounts genutzt, was mit einigen Detaillösungen besser funktioniert als rein maschinelles Lernen.

Das 2011 vorgestellte Konzept von Yang et al. [15] erzielt mit einem schwellwertbasierten Verfahren, also eigentlich einem Prinzip des maschinellen Lernens, bessere Trefferquoten als zuvor. Dabei erkennt es vor allem auch für graphbasierte Verfahren unauffällige, da stark vernetzte, Accounts. Dies wird nur durch neue Metriken realisiert, die scheinbar besser geeignet sind als die zuvor genutzten.

Daher ist es schwer, einen klaren Favoriten aus den Verfahren zu wählen. Die Kombination aus mehreren Verfahren mit besserer Prävention der ersten Kompromittierung von Accounts ist dabei das Beste, der Aufwand dieser Maßnahmen muss dabei auch immer berücksichtigt werden.

In Zukunft wird der Kreislauf des Angriffs und der Abwehr weitergehen und es müssen neue Verfahren, Metriken und Konzepte entwickelt werden, um die Angreifer zu schwächen. Die hier vorgestellten Möglichkeiten sind dabei schon sehr mächtig und derzeit als Hilfsmittel stark genug, um sich gegen Angreifer zu wehren. Dabei dient der Mensch als Endkontrollorgan, um die wahrscheinlichkeitsbasierten Treffer zu bestätigen oder abzulehnen. Die Ergebnisse können dann in die Bewertungskriterien zurückfließen, um die Erkennung zu verbessern.

8. LITERATUR

[1] Balduzzi et al.: *Abusing social networks for automated user profiling*, In Proceedings of the 13th International Symposium RAID, Ottawa, Ontario, Canada, Seiten 422-441, Springer-Verlag Berlin Heidelberg, 2010

[2] Benevenuto, Fabricio, et al.: *Detecting spammers on twitter*, Collaboration, electronic messaging, anti-abuse and spam conference (CEAS), Artikel 12, 2010

[3] Bonneau, Anderson, Danezis: *Prying data out of a social network*, ASONAM'09. International Conference on Advances in. IEEE, Seiten 249-254, 2009

[4] CAO, Qiang, et al.: *Aiding the detection of fake accounts in large scale social online services*, In Proc. of NSDI , 2012

[5] Danezis, George, and Prateek Mittal: *SybilInfer: Detecting Sybil Nodes using Social Networks*, In NDSS, 2009

[6] Gao, Hongyu, et al: *Detecting and characterizing social spam campaigns*, In Proceedings of the 10th ACM SIGCOMM conference on Internet measurement, Seiten 35-47, ACM, 2010

[7] Ghosh, Saptarshi, et al: *Understanding and combating link farming in the twitter social network*, In Proceedings of the 21st international conference on World Wide Web , Seiten 61-70, ACM, 2010

[8] Hong, J.: *The state of phishing attacks*, In Communications of the ACM, Seiten 74-81, ACM, 2012

[9] Irani, D.,Balduzzi, M., et al.: *Reverse social engineering attacks in online social networks*, In Detection of Intrusions and Malware, and Vulnerability Assessment, Seiten 55-74, Springer Berlin Heidelberg, 2011

[10] Stein, T., Chen, E., Mangla, K.: *Facebook immune system*, In Proceedings of the 4th Workshop on Social Network Systems, Atrikel 8, ACM, 2011

[11] Stringhini, G., Kruegel, C., Vigna, G.: *Detecting spammers on social networks*, In Proceedings of the 26th Annual Computer Security Applications Conference, Seiten 1-9, ACM, 2010

[12] Tan, Enhua, et al.: *UNIK: unsupervised social network spam detection*, In Proceedings of the 22nd ACM international conference on Conference on information & knowledge management, Seiten479-488, ACM, 2013

[13] Wang, Alex Hai: *Detecting spam bots in online social networking sites: a machine learning approach*, In Data and Applications Security and Privacy XXIV, Seiten 335-342, Springer Berlin Heidelberg, 2010

[14] Wang, Alex Hai: *Don't follow me: Spam detection in twitter*, In Proceedings of the 2010 International Conference on Security and Cryptography (SECRYPT), Seiten 1-10, IEEE, 2010

[15] Yang, Zhi, et al.: *Uncovering social network sybils in the wild*, In Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference, Seiten 259-268, ACM, 2011

[16] Zhang, X., Zhu, S., Liang, W.: *Detecting Spam and Promoting Campaigns in the Twitter Social Network*, In Proceedings of the 2012 IEEE 12th International Conference on Data Mining, Seiten 1194-1199, IEEE, 2012

[17] Google Inc. reCAPTCHA, www.google.com/recaptcha/, Stand 14.06.2014

[18] Google Watch Blog Google nutzt Streetview-Fotos für reCAPTCHA, `http://www.googlewatchblog.de/2012/03/google-streetview-fotos-recaptcha/`, Stand 14.06.2014

[19] Bundesministerium für Arbeit und Soziales: *CAPTCHAs und Barrierefreiheit*, `http://tinyurl.com/ngmfz8c`, Stand 14.06.2014

[20] *Google Explores +1 Button To Influence Search Results*: `http://tinyurl.com/7g927oy`, 2011, Stand 14.06.2014.

[21] *Nigerianische Betrüger nutzen Internet*: `http://tinyurl.com/n9hqg2l`, 2006

[22] *Google+ Account Suspensions Over ToS Drawing Fire*: `http://tinyurl.com/5vrt524`, 2011, Stand 14.06.2014

[23] *Google-safe-browsing: Protocolv2Spec*: `http://tinyurl.com/64rssm`, 2009, Stand 20.07.2014

[24] *How does built-in Phishing and Malware Protection work?*: `http://tinyurl.com/q8oxvcf`, Stand 20.07.2014

[25] Polizei-beratung.de: *Vorauszahlungsbetrug*: `http://tinyurl.com/mu9xqyz`, Stand 20.07.2014

[26] heise.de: *Spam feiert 30. Geburtstag*: `http://tinyurl.com/mu9xqyz`, 2008, Stand 20.07.2014

Self-Healing in Self-Organizing Networks

Oliver Scheit
Betreuer: Tsvetko Tsvetkov
Seminar Innovative Internettechnologien und Mobilkommunikation SS2014
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: oliver.scheit@tum.de

ABSTRACT

Self-Organizing Networks are a way to manage the increasing number and complexity of current mobile networks. They provide functions to automatically configure network elements allowing for faster installation of new network elements. The domains SON also include the ability to self-optimize the network parameters and to detect and solve problems. The detection and diagnosis of degraded network elements is based on performance indicators provided by the individual network element. The diagnosis furthermore requires the combination of the knowledge of the correlation between the performance indicators and the corresponding faults. This paper shows an example framework that implements the detection and diagnosis of faults and provides an example of a SON-function that can find conflicts during the self-automated processes.

1. INTRODUCTION

Due to increasing complexity seen in today's networks, methods to efficiently manage them have to be introduced. One way to do so is to automate various parts of the system and create a so called "SON", short for self-organizing network. The goal of a SON is to configure, maintain and self-heal parts of the system based on information the system gathers itself, without needing manual assistance from the systems operator.

One usage area of SONs is mobile communication, in particular LTE (long term evolution), where a large amount of Base Stations are required for coverage purposes. Deploying a bigger number of network elements however means that there is a higher chance of failure in the systems network. This is the reason self-healing of self-organizing networks was introduced as a requirement for LTE by 3GPP, the 3rd Generation Partnership Project, to automatically identify faulty behavior.

This paper aims to give a short summary about the general idea and the domains of self-organizing networks and a more detailed overview about the self-healing aspect of SONs based on LTE network deployment. This includes the detection of a faulty element and the diagnosis of the root problem.

The document is structured as follows: in Section 2 self-organizing network domains will be explained and the benefits of automation will be shown. Section 3 will detail the general self-healing idea. Section 4 gives an insight into detection of degraded network elements and Section 5 will show different ways to diagnose the root-cause of problems in the network. Section 6 will show case how to implement an automatic self-healing framework. In Section 7 a SON-function to detect problems in the self-optimization process is outlined and Section 8 shows other possible uses for SONs. Section 9 will then conclude the paper.

2. SELF-ORGANIZING NETWORKS

Modern mobile communications networks tend towards end nodes with smaller range to be able to serve users with higher bandwidth and more stable connections, this drastically increases the amount of network elements to be deployed and maintained. Configuring, optimizing, monitoring and troubleshooting all individual cells therefore is an unpractical, if not impossible, task for the networks operator. To provide an efficient cost-effective network these tasks have to be at least partly automated so that the operator only has to supervise the parameter provided to him by the system and the SON processes. Such a self-managing network is commonly referred to as a self-organizing network. In mobile communications the domains of a SON include self-configuration, self-optimization and self-healing of the systems elements. [3]

2.1 Self-Configuration

To be able to efficiently install new hardware or software the deployment of network elements should be a *"plug & play"* system in which cells automatically download the necessary setup information and adapt to their environment. The SON self-configuration enables the system to add new elements to the existing network in real-time. The auto configuration sets up physical parameters like Physical Cell Identifier, transmission power and antenna tilt. In addition it is responsible for neighbor discovery and handover parameters. [3]

2.2 Self-Optimization

The goal of self-optimization is to find the most efficient parameters for a network at any given time. If the network environment changes over time, the system has to reconfigure parameters accordingly.[3] Some examples of self-optimization are

- Mobility Load Balancing (MLB) which enables cells to redirect part of the user generated traffic to neighboring cells. This allows the load to be evenly distributed across cells giving customers a better throughput and allowing an improved user experience. MLB also increases the overall capacity of the network and helps avoiding a congested cell while neighboring cells are unused.
- Energy saving allows cells to be deactivated during times of low traffic to reduce the power consumption of

doi: 10.2313/NET-2014-08-1_22

the network without impairing the overall network performance.

- Mobility Robustness Optimization or MRO. This function aims to minimize intra-LTE handover failures or unnecessary handovers to other radio access technologies. Typical scenarios include handovers initiated too early/late or to wrong cells.
- Inter Cell Interference Coordination enables cells to coordinate their signals to decrease interference i.e. cell A uses the lower part of the channel bandwidth while cell B uses the upper part

All of the functions above can be implemented as decentralized functions in each network element. [2]

3. SELF-HEALING

Self-healing is the least researched of the self-organizing functions as it is the most complex of the SON domains. Nevertheless it is still an important part of self-management as even though the network won't be able to recover a physical damaged cell, it can help identify which network element is not functioning as expected. The self-healing processes should contain:

- Alarm correlation, a diagnosis process triggered on alarms that tries to find the root cause of the alarm. If an possible cause is discovered, an automatic recovery action can be started to try to resolve the problem
- Sleeping cell or cell degradation detection, locate cells that don't transmit faults/alarms but still do not perform as planned
- Cell Outage Compensation; reconfigure neighbor cells temporarily to compensate the failure of another cell.

3.1 Self-healing use cases by 3GPP

The 3GPP, 3rd Generation Partnership Program has therefore defined sell-healing use cases.

Self-recovery of Network Element software: try loading an older software version or configuration to solve faults. This can be done multiple times until you reach a certain amount of tries. The process will then inform the IRP-Manager if the recovery was successful or not.

Self-healing of board faults: discovery of malfunctioning network element hardware. If there is a redundant hardware element the cell will try to activate the backup. If there is no redundancy and there is a loss of radio services, cell outage management will be started. The use cases for cell outage management include:

Cell outage Detection: The System detects a degraded, out-of-service or a "sleeping cell". A cell that does not serve user equipment with requested data even though users are connected to the cell and that does not raise any alarms to indicate the malfunction. To detect these kind of problems so called performance indicators are observed and irregularities are reported to the operator.

Cell outage Recovery: Based on the diagnosis an available recovery action is performed to restore the systems normal capabilities and the results will be reported.

Cell Outage Compensation: as stated above, the network reconfigures itself to compensate for potential coverage holes created by the failure.

Return from Cell Outage Compensation: After the fault has been resolved the cells that took part in the compensation process have to return to their pre-fault state to ensure the optimal configuration state.

3.2 3GPP self-healing process

The proposed general self-healing process is structured as follows: An input monitoring function continuously checks whether a performance indicator has violated its corresponding threshold. In case one or more did the associated self-healing process will be triggered. The process will then gather additional system information like performance indicators, system variables, test results and radio measurements. See chapter 4 for a more detailed explanation about the detection process. With this aggregated data an analysis function is started to detect the root cause of the problem. Depending on the result of the diagnosis, i.e. was a there a problem at all, corrective action is taken if the problem can be solved through a SON-function. It is advisable to back up the system configuration prior to executing the SON process however. Afterwards an evaluation of the new system state is initiated to see whether the problem has been resolved or not or even if the new configuration created new problems. These steps are repeated until the fault is solved or stop conditions i.e. a reset counter is fulfilled. The result of the self-healing process will be monitored by the Self-healing and Monitoring function that has access to data of all of the above mentioned process and allows the operator to control the execution of the self-healing process. [3]

3.3 Cell degradation management

Cell Degradation management is one of the most important parts of the self-healing process. Detection and Diagnosis of faults can reduce the costs for the operator and help improving the user experience.

The typical problems found in cellular networks are hardware or software faults, planning and configuration faults and environmental changes. Without SON functionality the degradation detection is only partially automated. The system will check itself for alarms, once an alarm is found the most common attempt to fix the cell is to reset it. If the problem is not resolved after a given number of resets the system will alert the operations department. The element will then be investigated remotely, in case this does not help solve the problem a technician will have to inspect the cell on-site and possibly replace hardware elements or the whole cell. Not all cell degradations cause alarms however, to see if a cell is working in an efficient, optimal way the optimization department will have to assess additional parameters provided by the cell. These are called KPI or PI, (key) performance indicators. To find malfunctioning cells thresholds for these parameters are set that should not by violated i.e. percentage of dropped calls. Base stations that breach the allowed limit or that show the biggest change in performance from one time period to another will then be selected for further investigation. [3]

doi: 10.2313/NET-2014-08-1_22

3.4 Cell outage compensation (COC)

Cell outage compensation is one of the most crucial parts of the self-healing use cases. After the detection and diagnosis of a faulty network element neighbor cells will be configured to partly provide coverage for the area of the broken cell. The COC will be triggered when it is clear that the loss-of-service will persist for a prolonged period of time. To be able to do this several parameters of the neighboring nodes can be tweaked like antenna tilt, transmission power, beam forming and shaping. Since it is desirable to recover the service in the area as fast as possible to minimize user complaints the COC process will not try to find optimal parameters and settle for providing coverage in the affected area as fast as possible. After the COC process is finished additional self-optimization algorithms can be executed to find optimal settings for the new network environment. [3]

4. DETECTION OF DEGRADATIONS

The detection of faulty hardware or performance degradations plays an important role in the self-healing domain. Being able to reliantly detect problems in cells allows for faster diagnosis and reduces human resources. The detection algorithm does not equal the diagnosis process. Detection focuses on finding anomalous behavior and triggers an alarm if it found an unusual system state. The diagnosis process then gathers the information provided by the detection and concludes whether this is an actual fault or normal system behavior. The detection of degradation is based on event counters, KPIs and alarms. The process will determine if the provided performance indicators are in an acceptable range i.e. “healthy”. As these performance indicators are usually stochastically distributed statistical methods to define a healthy state have to be found. To do so you can create profiles for each of the KPIs that describe a range of normal, expected behavior. A KPI that leaves this range could point to degradation in the network performance. This approach has the advantage that the operator can define states that are normal and does not have to define states that are degraded, as these are often hard to define and statistically seen rather rare in overall network lifetime. The profiles can be split into three basic categories:

- Absolute threshold: this profile defines a threshold that should not be violated. Therefore the KPI should not exceed or fall short of. Examples for values that should not be exceeded are fault or failure rates like call drop ratio. An example for a rate that should not fall below a certain threshold would be the handover success ratio. See Figure 1
- Statistical profile: This profile defines an acceptable upper, maximum threshold that should not be exceeded and a lower, minimum threshold. These profiles can define a mean and it corresponding standard deviation that should not be violated. This profile can be symmetric or asymmetric. See Figure 2
- Time dependent profile: Much like the statistical profile there is an upper and a lower threshold that the KPI should comply with. The difference is that these thresholds fluctuate with time, meaning that acceptable borders will change during the time period. These profiles are typically user driven, e.g. traffic generated in the system. See Figure 3

While most of the task mentioned above, are already done automatically, the process to define what is faulty behavior and what is normal, but until now not encountered behavior has to be further refined to improve the accuracy of the fault detection. There are two different approaches for the anomaly detection. The first is the univariate, here a single performance indicator is observed and its specific distribution is considered to decide if it is in an acceptable range of its past mean and deviation.

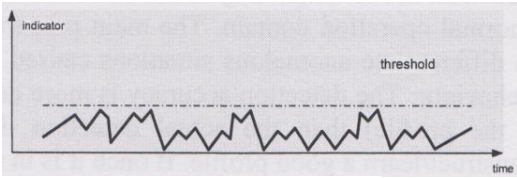


Figure 1: absolute threshold [3]

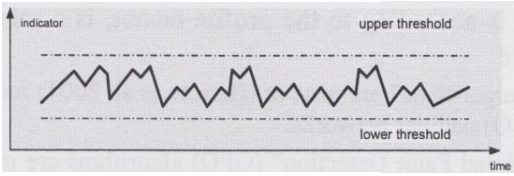


Figure 2: statistical profile [3]

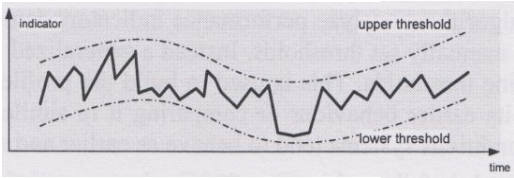


Figure 3: time dependent profile [3]

A drawback of this approach is that unusual behavior can be, depending on the profile, declared as a faulty behavior even though the cell is working fine. The other possible method is the multivariate anomaly detection where, as the name suggests, multiple performance indicators are considered and compared against each other to determine if the cell is working correctly. In this approach the performance of the cell as a whole rather than single KPIs can be evaluated, which makes it more reliable at avoiding false alarms.

The quality of a detection algorithm can be classified by several grades, as shown in [3]:

- The first is the detection accuracy, which gives an indication about the reliability of the detection algorithms. The goal of the detection process should be to have a high level of true negatives, i.e. no fault no alarm and true positives, i.e. a fault has occurred and an alarm was triggered and to avoid false negatives or false positive, i.e. an alarm was trigger even although there was no fault or worse there was a degradation but it was not detected by the process.
- Detection delay is the amount of time passed between a fault appeared and the detection of it. The faster the detection works, the faster the problem can be resolved or at least compensated.
- Other relevant metrics for operators may include severity indication accuracy that gives the operator an

doi: 10.2313/NET-2014-08-1_22

estimation of the overall severity of the degradation so that he may schedule maintenance accordingly. The detection process should give a reliable estimate of the severity, because simpler less performance impairing faults do not have to be resolved immediately, whereas grave faults like cell outage might need fast corrective action. Signaling Overhead describes the network load produced by the detection process between the individual cells, the cell and the network management and the cell and user devices, e.g. User equipment measurements.

- The last metric is the Processing Overhead, i.e. the amount of processing time needed for the detection algorithm. This typically depends on the amount of input data and can be exponential which poses a problem for network elements small processing capabilities.

5. CELL-DEGRADATION DIAGNOSIS

While the detection of problems in the network is at least partially automated nowadays the diagnosis of the corresponding root cause is still a manual task carried out by troubleshooting experts with years of experience. The general troubleshooting process starts with a notification of the expert by the detection system. The experts will then gather additional performance indicators and identify possible root causes for the problem. In case the problem can't be identified by these means a drive test will have to be performed on-site. An increase in network elements however makes manual troubleshooting a time intensive and cost expensive task for operators.

The complete automation of this process is the key goal of self-organizing networks. A self-healing system would require less maintenance and less troubleshooting personnel to manage bigger systems than before.

The challenge is to build a system that, like a human, can make decisions based on prior knowledge and can use its observations to include new similar cases to its database. To do so the system will have to learn a few basic symptom – root cause relations from existing human knowledge, this phase is called the learning phase of the diagnosis system. The next phase would be the assistive phase, where the diagnosis process would inform the troubleshooting expert about the most probable cause of the problem, but the execution of recovery action would still be done by the human operator. The root cause diagnosis will then be fine-tuned by knowledge of the troubleshooting expert. The last phase would be a completely autonomous system where the human operator only supervises decisions done by the self-healing process.

The critical component of this system is mapping symptoms to their respective root cause in an efficient, but reliant way. There are several ways to implement this association; some examples will be explained in short in the following sections, a more detailed view on these can be found in [3].

5.1 Rule Based Systems

The easiest way to implement this diagnosis is the rule based system. Every set of symptoms A is mapped to the root case B in a simple *"if (A) then (B)"* style of rules. Obviously a rule based system is easy to train as new fault cases can added as a new rule set. This system is very similar to human reasoning and easy to understand. It also proved to work efficiently in smaller deterministic environments, but the huge number of different parameters in mobile networks and their linked nature make these systems hard to fine-tune to effectively find problems in mobile networks. Additionally symptoms cannot always be associated to a single root cause, but with a probability. Rule based system only allow one specific case for a set of input parameters.

5.2 Bayesian Networks

Based on Bayes' theorem this system is able to include the uncertainty included in the performance indicators. Every root cause can be mapped to different symptoms and every symptom can be linked to different root cases. These links however do not always appear with the same frequency, thus they can be expressed with probabilities. The whole Bayesian network can be visualized by an acyclic directed graph to give a better overview of the individual dependencies for the operator. Since the direction of the tree does not matter operators can build the graph from the top-down. This means he can define new root-causes and add the accompanying symptoms, turning this into an easy to maintain and improve diagnosis system. The problem with this model however is that for each parent node several probabilities for its child nodes have to be defined. These probabilities are not always obvious or can be determined with the required knowledge for the system to work with the desired accuracy. Furthermore with a growing number of nodes the size of the conditional probability table grows exponentially in its size making it inefficient to compute and maintain. The Bayesian model can be simplified by two different models one is the naïve model, where only one root cause can be present each time. This simplifies the model because symptoms cannot be associated with multiple root causes which in turn decreases the amount of processing power required. The noise-or model allows multiple root-causes. It reduces the amount of space needed for the conditional probability table from 2^n to n by representing every root cause as a binary node that either exists or not. Than the system will measure the individual contribution of the root cause to each symptom.

5.3 Case Based Reasoning

Bayesian Networks heavily rely on the distribution of faults to determine the root-cause, but during normal operation of a network element these fault cases should be rather rare. This makes finding a correct probability for the conditional probability table nearly impossible as reliable data on faults cannot be provided or the statistical sample is so small that even minor deviations can trigger false alarms. Case Based Reasoning therefore tries to classify normal operation and will only react if there is a deviation from this "healthy" state of the cell. The basic idea behind a CBR system is to continuously build on the fault states already discovered by the system. The system starts with no knowledge of fault cases and monitors the performance indicators for anomalies, if an anomaly is found it tries to solve the problem by looking at previously encountered fault states with identical KPI levels and tries to solve the problem accordingly to previous solutions. If the root case was found and the problem was solved the system adds this new case with the associated parameters to its fault database. In case the problem was not resolved the system informs an operator to troubleshoot the cell. The result of this manual troubleshooting process will then be added to the existing cases creating an ever growing database of root cases with their respective symptoms.

doi: 10.2313/NET-2014-08-1_22

6. AUTOMATIC DETECTION AND DIAGNOSIS FRAMEWORK

This Section will give a closer look into the working of an automatic detection and diagnosis framework proposed by Nováczki et al[4] and the exact mechanics behind the system. The system should be able to integrate into existing network environments without having to perform major modifications in the existing network. Additionally it has to be able to adapt itself to the existing system and its performance indicators and other available parameters. It is similar to case based reasoning, with the difference that it does not try to resolve issues by comparing it directly to older cases, but by assigning KPI footprints to so called targets and comparing these reports with the current measurements gathered by the system.

6.1 Detection

Instead of defining thresholds for variables that generate a binary result for the diagnosis algorithm, i.e. fault yes or no, a way to transform the KPI value into a continuous number in the range from 0 to 1 is proposed. Where 0 means perfectly healthy behavior and the 1 is asymptotically approached as the KPI state decreases. To build a profile of the KPI n consecutive samples x_i of the KPI are averaged to build one sample a_i. The KPI variable corresponding to the performance indicator K is called K^*. The profile will then be built using the mean and variance of the K^*. Assuming the x_i are independent and identically distributed the variable K^* should be normally distributed with

$$K^*_{n\to\infty} \sim \mathcal{N}(\mu(K^*), \sigma^2(K^*))$$

As a general example n should be bigger than 40 and more than k > 20 samples should be used, if the samples are not correlated. If the samples are correlated more samples are needed. After the profiling process has been completed, the current state of the KPI called $X^{(K)}$ is computed using the average of the latest n samples of the variable K^*. This current mean $X^{(K)}$ can then be transformed into a standard normal variable

$$Z^{(X)} = (X^{(K)} - \mu(K^*)/\sigma(K^*))$$

Where μ is the mean of the variable, this means the expected value and σ is the root of the variance the variable shows. The variance is the expected deviation from the mean.

After creating the profile K^* and current state $X^{(K)}$ of the KPI, a level function to evaluate the state and transform it into a number in the range from 0 to 1 has to be defined. The level of the KPI will be denoted by $\varphi(K)$. The evaluation will be based on the desired thresholds of the KPI profile, i.e. upper threshold, lower threshold or both and the number of standard deviations -C allowed to result in the KPI level of 0.5. An acceptable upper threshold will be $\varphi_+(K)$ meaning that the KPI can decrease but should not increase. $\varphi_-(K)$ is the opposite where an increase in KPI levels is acceptable. Finally $\varphi_\pm(K)$ where neither an increase nor a decrease of the KPI level is desirable. The resulting functions are:

$$\varphi_+(K) = \Phi\,(C + Z^{(K)})$$

$$\varphi_-(K) = \Phi\,(C - Z^{(K)})$$

$$\varphi_\pm(K) = \varphi_+(K) + \varphi_-(K)$$

Where Φ is the cumulative distribution function of $\mathcal{N}(0,1)$, i.e. the standard normal distribution. The advantage of this approach is that the function does not check for a single violation of a threshold, but for a constant continuous decrease in the overall KPI performance. It also provides a uniform level function for all KPIs without the operator setting specific thresholds for single KPIs. The problem with this definition of a profile however is that most performance indicators are user behavior dependent, meaning that the profile has to adapt to different user behavior in different time periods of the day. Using a profile with a large standard deviation proves impractical as it impairs the detection process, therefore constructing different profiles for different time slots should be created. Too get a mean of a specific time t you can interpolate the means of the times t_1 and t_2 using a standard linear interpolation function. [4]

6.2 Diagnosis

Now that the detection of KPI levels has been defined the actual mapping of the levels to Targets has to be created. Targets can be corrective actions or root cases. This mapping is called a report. Each time a fault with the same symptom appears an identical report will be created and added to a database so that the frequency of a target can be measured as well. These reports can either be created by an expert for the system or added based on the solution of previous faults occurring in the network. Each report contains the subset of KPIs that were observed and the target that was analyzed by the specific report. There is a special target T_0 which means that the system is working perfectly healthy and its report R_0 that contains an empty set of KPI and the target T_0. No other report however is allowed to have an empty set of KPIs.

In order to find targets the system continuously monitors the KPIs and the "winning" target is the one that matches the most KPI states based on reports previously acquired by the system. Because not all KPIs are relevant to a target and not all reports for a target have the same KPI subset a likelihood function is introduced to evaluate which performance indicators are actually relevant to the target. This likelihood function is denoted as $l^T(K)$ and shows the relevance of the KPI. It is calculated by dividing the number of reports for target T that contain the KPI K through the overall number of reports for T. A likelihood of 0.5 means that the observed KPI was present in half of the reports for T rendering it useless as an indicator for T, because it does not appear to be linked to the target.

This can be further refined by balancing the KPI with a consistency function $cf(x) = 2 * |x - 0.5|$. This consistency function gives a weighted relevance of the KPI to a target. If the KPI K likelihood is close to 0 or to 1 for a Target T_i the KPI will have a high relevance, since you can say that whenever K is absent or present the target T_i is present too. To be able to compare different targets you have to quantify a score that is dependent on the KPIs. The score of a single KPI s(K) towards a the Target can be realized by multiplying the consistency function with a hamming distance:

$$d^T(K) = \varphi(K) \text{ if } l^T(K) \geq 0.5 \text{ , else } 1 - \varphi(K)$$

The overall score of a target is the score of all KPI scores for T:

$$S(T) = \sum s(K_i).$$

In case two targets T_1 and T_2 end up with the same overall score, the target with the higher overall number of reports will get the priority.

Due to the consistency contradictory reports should be checked as they can cause the KPI scores to become close to zero giving the

doi: 10.2313/NET-2014-08-1_22

target a lower score in the future. Contradictory results may also be an indication that there is an unknown error that should be investigated and added to the database. [4]

6.3 Integration into the Network

In general not all KPIs will be kept live to keep the amount of data an element creates to a minimum level, therefore not all KPIs should contribute to the initial detection of a problem. These inactive KPIs however could contain relevant data to the solution and need to be considered during the diagnosis. To resolve the problem an update of these KPIs is needed and should be initiated by the system. This is called an active measurement; they can be useful if a conclusion between different targets cannot be reached by the initial diagnosis process. In case the active measurement can't be initiated by the system itself it should notify a troubleshooting expert with an alarm that an active measurement is required to resolve a problem. To find the most suitable KPI that will most likely resolve the stalemate the KPI with the maximum likelihood for one of the two cells should be considered, as it will most likely be the most impacting on the overall score of the respective targets.

One advantage of the proposed framework is that it can, identical to case base reasoning, start with an empty knowledge base on targets and add targets and their corresponding reports during its lifetime. This means that the operator only has to initially do the troubleshooting process and can let the system take over if it has enough targets and reports to guarantee optimal operation. The management of the expert knowledge generated by the system could be stored on a management server and set to synchronize regularly with the cell to keep the information up-to-date. It would also enable the troubleshooting expert to edit the reports and add a root cause to help the system. [4]

6.4 Possible improvements to the system

To further reduce the amount of human interaction required by the system an extension to this system can be made where the operator only has to specify time slots where normal operation was detected. The framework can then implement a so called two-sample Kolmogorov – Smirnov test, that can compare two random distributions to check whether they have the same distribution or not.

Therefore it compares the empirical cumulative distribution functions, ECDF, and computes the maximum distance between these two functions. See Figure 4. [5]

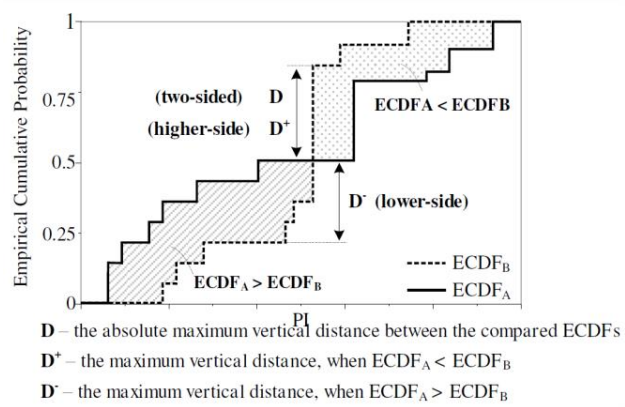


Figure 4: ECDF – comparison [5]. D is max{D^+;D^-}, D^- is the max distance of $ECDF_A > ECDF_B$. D^+ the max of $ECDF_B > ECDF_A$

The ECDF is the cumulative addition of the probabilities over all events. Therefore the ECDF ranges from 0 to 1.

The distribution of a KPI profile is its corresponding ECDF function, which means that an ECDF will have to be computed for every KPI that will be considered in future diagnosis processes. The operator now only has to mark the time slots in which the KPIs and the overall system operation were considered "healthy". The input is then divided into fragments. For each of these fragments the ECDF is computed and saved as a possible profile candidate. To reduce the number of profiles only the most representatives are stored however. Then a process starts to find the minimum number of profiles that cover all the profile candidates. If a profile covers a candidate is decided as presented earlier by the two-sample Kolmogorov-Smirnov test. [5]

To detect if one PI shows unusual behavior its latest N samples from the observation window are taken and the Actual-ECDF of the PI is computed. The detector then computes the maximum distance of this A-ECDF from all the profiles in the database of this particular PI. The minimum distance will be considered as the best matching example and it will be compared to a similarity threshold. Distances below this threshold can be defined as "healthy" operation, since fluctuation in parameters is normal to a certain level. Since there are different thresholds for each PI different methods to compute the distance will be used.

- Success Indicators: if the values are higher than expected there is no fault present to therefore the distance of the lower side will be use D^- (Figure 4)
- Failure Indicators: Values should not fall below a certain threshold. The relevant distance is the high sided D^+
- Neutral: Values should be between a certain range which means that both distances will be considered in diagnosis

An unknown distribution can either mean normal operation that differs from all profiles gathered before or the system is in a faulty state. This has to be decided by the operator that marks this new distribution as a new error or normal behavior. [5]

7. SON-FUNCTION COORDINATION

Several SON-functions might interfere with each other at run-time, e.g. the SON MRO (mobility robustness optimization) and the CCO (coverage and capacity optimization) where the MRO might try to resolve handover failures, but these failures are caused by a coverage hole, for which the CCO function would be responsible. To prevent this hierarchies of priorities could be introduced for SON-functions, but this could lead to deadlocks where a high priority function which is unable to fulfil its goal blocks a low priority function that could have resolved the problem. Another approach however is to implement another SON-function that monitors the execution of the other functions and tries to find locks in which no improvement in the network is seen, even though parameters are constantly adjusted. This function can be called a SONOT (SON Operational Troubleshooting) that has two main aims, to monitor the execution of other functions and find unresolvable problems and to help resolve the problem automatically.

To detect a problem with a SON-function different approaches can be used a state-based approach that only considers the current system state and a history based approach that allows considering

doi: 10.2313/NET-2014-08-1_22

the influence of changes to parameters to the overall system performance. It also enables the system to predict whether a function will be able to fulfil its target or not. It can also track if a function continuously tweaks parameters, an indicator that it is not able to achieve its desired state.
The SONOT function will then try to analyze the functions involved and the root cause of the problem. It will also try to block all non-relevant SON-function execution to block any interference into the self-healing process. It will then call an more suitable SON-function to resolve the problem or if the problem is not resolvable by the system it will notify the network operator that the problem cannot be solved manually and requires manual troubleshooting. [1]

8. Other Self-Organizing Systems

Of course the applicability of SONs is not limited to mobile communications, another example of this network type is the freifunk network that is currently in development in Germany and Austria. This project tries to create an open network that provides internet access to all participants. To do so the nodes have to be able to find a path to an internet distributing node and deal with constant changes to the network topology in a self-organizing manner [6].

A different application from wireless networks would be road networks. As shown in [7] by Prothmann et al. local, distributed routing decisions can be applied to improve the traffic flow in cities in the future. The paper shows that the routing algorithms used in the internet like Link State and Distance Vector can be modified to use them to control traffic. The traffic light at intersections equal routers and can be used to control how many vehicles can pass through a given route and optimize time to a cities important locations.

9. Conclusion

Self-Organizing Networks are an efficient way to manage the increasing complexity in cellular networks seen today. They offer an easy way to install new cells for the operator with their plug & play style self-configuration and reduce the amount of maintenance required through their self-optimization capabilities. At the same time they possibly improve the user experience beyond a level possible by manual configuration of parameters. As described in this paper there are also ways for the system to automatically deal with problems occurring during the run-time of the system. This reduces the amount of time needed to detect problems in the network and can decrease the amount of necessary drive tests in the network. It can also help detecting problems more reliably than it would be possible with manual troubleshooting. This paper also showed the requirements needed like reliable performance indicators and the corresponding profiles. The different diagnosis approaches are introduced and their respective weaknesses and advantages are explained. Then a conceptual framework is showcased and the way it handles the detection of degradation and how it build its profiles is explained. The diagnosis process and its advantages against other similar approaches are shown. In the last Section an approach to fault handling of SON-functions is outlined that can detect problems in the interaction of SON-functions or find functions that cannot reach their desired objectives. Other applications of Self-Organizing Systems are then given as a conclusion to the paper.

10. Literature

[1] Christoph Frenzel, Tsvetko Tsvetkov, Henning Sanneck, Bernhard Bauer, and Georg Carle. Detection and Resolution of Ineffective Function. Behavior in Self-Organizing Networks. In IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM), Sydney, Australia, June 2014.

[2]Chris Johnson. Long Term Evolution IN BULLETS. Edition 2, version 1. ISBN 9781478166177. Amazon Distribution GmBH, Leipzig, 2012

[3]Seppo Hämäläinen, Henning Sanneck, Cinzia Sartori. LTE Self-Organising Networks (SON): Network Management Automation for Operational Efficiency. ISBN 9781119970675. John Wiley & Sons, 2012.

[4]Péter Szilágyi, Szabolcs Nováczki. An Automatic Detection and Diagnosis Framework for Mobile Communication Systems .IEEE transactions on network and service management, VOL. 9, NO. 2, JUNE 2012.

[5]Szabolcs Nováczki, S., "An improved anomaly detection and diagnosis framework for mobile network operators," Design of Reliable Communication Networks (DRCN), 2013 9th International Conference on the, vol., no., pp.234,241, 4-7 March 2013

[6] http://freifunk.net/, accessed 29.07.2014

[7] Holger Prothmann, Sven Tomforde, Johannes Lyda, Jürgen Branke, Jörg Hähner, Christian Müller-Schloer, Hartmut Schmeck. Self-Organised Routing for Road Networks. Self-Organizing Systems. 6th IFIP TC 6 InternationalWorkshop, IWSOS 2012 Delft, The Netherlands, March 15-16, 2012 Proceedings. ISBN 978-3-642-28582-0

ISBN 3-937201-43-2
DOI 10.2313/NET-2014-08-1

ISSN 1868-2634 (print)
ISSN 1868-2642 (electronic)