SDN-Enabled Resiliency, Monitoring and Control in Computer Networks

Jonathan Vestin

Faculty of Health, Science and Technology

Computer Science

DOCTORAL THESIS | Karlstad University Studies | 2020:2

SDN-Enabled Resiliency, Monitoring and Control in Computer Networks

Jonathan Vestin

DOCTORAL THESIS | Karlstad University Studies | 2020:2

SDN-Enabled Resiliency, Monitoring and Control in Computer Networks

Jonathan Vestin

DOCTORAL THESIS

Karlstad University Studies | 2020:2

urn:nbn:se:kau:diva-75953

ISSN 1403-8099

ISBN 978-91-7867-074-1 (print)

ISBN 978-91-7867-075-8 (pdf)

Distribution:
Karlstad University
Faculty of Health, Science and Technology
Department of Mathematics and Computer Science
SE-651 88 Karlstad, Sweden
+46 54 700 10 00

Print: Universitetstryckeriet, Karlstad 2020

WWW.KAU.SE

SDN-Enabled Resiliency, Monitoring and Control in Computer Networks

Jonathan Vestin
Department of Mathematics and Computer Science

Abstract

Next generation networks aim to increase network convergence by allowing a single network architecture to serve diverse traffic types ranging from high-bandwidth video streaming to low-latency industrial automation, while meeting their respective service level requirements. Such a converged network architecture puts high requirements on flexibility, interoperability, and resilience. While current networks exhibit some degree of network convergence, they may not reach the level of interoperability required for future application areas. This is particularly prevalent in networks that depend heavily on closed and proprietary equipment, such as industrial automation and small cell backhaul networks. Recently, Software Defined Networking (SDN) and Network Function Virtualization (NFV) have been proposed as solutions for increased network flexibility. By separating and logically centralizing the network control plane, SDN allows for dynamic control of the network infrastructure. NFV, on the other hand, enables flexibility and scalability through the virtualization and orchestration of network functions.

In this thesis, we investigate how SDN and NFV can be used to make next generation networks more reliable, flexible and programmable. We focus mainly on three areas: resiliency, monitoring, and control. First, we look at the usage of SDN to enable in-network resiliency in wireless access, backhaul and industrial automation networks. Next, we design and evaluate FastReact, a switch program that allows industrial automation networks to partially offload their distributed application logic to the data plane, reducing end to end latency and increasing network resiliency. Finally, we propose combining FastReact control with in-network telemetry event detection, significantly increasing the monitoring capacity by selectively discarding redundant telemetry information in the data plane.

Acknowledgements

I am grateful for the support of my supervisor and mentor Andreas Kassler, and my co-supervisors Karl-Johan Grinnemo and Johan Åkerberg. Also thanks to Peter Dely and Martin Blom, both for encouraging my continued studies in computer science. Last, but not least, a special thank you to my family for supporting me during the writing of this thesis.

Karlstad University, January 7, 2020 Jonathan Vestin

Contents

List of Appended Papers

I. **Jonathan Vestin** and Andreas Kassler. QoS enabled WiFi MAC layer processing as an example of a NFV service. *2015 1st IEEE Conference on Network Softwarization (NetSoft)*, London 2015

Comments on Participation: I implemented the QoS extensions to Open vSwitch, which includes support for SFQ, CoDel, and FQ_CoDel. I also extended Open vSwitch to support classless (non-discriminatory) traffic control algorithms. I set up and performed the experiments, and analyzed the results. I am the main author of the paper.

II. **Jonathan Vestin**, Andreas Kassler and Johan Åkerberg. Resilient software defined networking for industrial control networks. *2015 10th International Conference on Information, Communications and Signal Processing (ICICS)*, Singapore 2015

Comments on Participation: I extended the CORE emulator to support Open vSwitch and added the traffic model for industrial automation networks. I set up the testbed, performed the experiments and analyzed the results. I am the main author of the paper.

III. **Jonathan Vestin** and Andreas Kassler. Low Frequency Assist for mmWave Backhaul - The Case for SDN Resiliency Mechanisms. *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, Paris 2017

Comments on Participation: I implemented the mmWave LOS/N-LOS/outage model in the CORE emulator. I set up the testbed, performed the experiments and analyzed the results. I am the main author of the paper.

IV. **Jonathan Vestin**, Andreas Kassler and Johan Åkerberg. FastReact: In-Network Control and Caching for Industrial Control Networks using Programmable Data Planes. *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, Torino 2018

Comments on Participation: I came up with the method for storing CNF logic statements in the P4 switch memory. I designed and implemented the P4 program for FastReact. I also performed the feasibility analysis, testbed setup, evaluation, and analysis of the results. I am the main author of the paper.

V. Deval Bhamare, Andreas Kassler, **Jonathan Vestin**, Mohammad Ali Khoshkholghi, and Javid Taheri. IntOpt: In-band Network Telemetry Optimization for NFV Service Chain Monitoring. *2019 IEEE International Conference on Communications*, Shanghai 2019

Comments on Participation: I designed and implemented the P4 program, designed the NetFPGA testbed, created the packet traces and performed the NetFPGA experiments. I contributed to writing and reviewing the paper, particularly focusing on the evaluation aspects.

VI. **Jonathan Vestin**, Andreas Kassler, Deval Bhamare, Karl-Johan Grinnemo, Jan-Olof Andersson and Gergely Pongracz. Programmable Event Detection for In-Band Network Telemetry. *2019 IEEE International Conference on Cloud Networking (CloudNet)*, Coimbra 2019

Comments on Participation: I implemented the `AF_XDP` program for INT event collection, part of the P4 INT Sink event detection switch program. I also designed, set up and performed the experiments. I am the main author of the paper.

VII. **Jonathan Vestin**, Andreas Kassler. In-Network Caching and Control for Industrial Automation Publish/Subscribe Networks. *In submission*

Comments on Participation: I implemented the P4 program, both for the Netronome Agilio P4 target and T_4P_4S compiler. I also implemented MicroC external functions, granular atomic lock extensions for T_4P_4S and the Python generator. I designed, set up and ran the experiments, and also analyzed the results. I am the main author of the paper.

Other Papers

Apart from the papers included in this thesis, I have authored and co-authored the following papers.

1. **Jonathan Vestin** and Andreas Kassler. QoS Management for WiFi MAC Layer Processing in the Cloud: Demo Description. *In Proceedings of the 11th ACM Symposium on QoS and Security for Wireless and Mobile Networks (Q2SWinet)*, Cancun 2015. (Demo Paper) **Best Demo Award**

   Relation to Included Paper: This paper presents the Open Daylight extensions, and integration with InfluxDB and Grafana to Paper I.

2. **Jonathan Vestin** and Andreas Kassler. Resilient SDN based small cell backhaul networks using mmWave bands. In *IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, Coimbra 2016. (Poster Paper)

   Relation to Included Paper: This paper includes a presentation of the emulation environment used in Paper III, and also includes preliminary evaluation results.

3. **Jonathan Vestin** and Andreas Kassler. SDN enabled resiliency in LTE assisted small cell mmWave backhaul networks. In *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, Paris 2017. (Poster Paper)

   Relation to Included Paper: This paper is a preliminary paper to Paper III, which focuses more on packet loss and latency.

4. **Jonathan Vestin** and Andreas Kassler. QoS Enabled WiFi MAC Layer Processing as an Example of a NFV Service. *Proceedings of Swedish Communication Technologies Workshop (Swe-CTW)*, Karlstad 2015. (Poster Paper)

   Relation to Included Paper: Poster version of Paper I

5. Rasmus Dahlberg, Tobias Pulls, **Jonathan Vestin**, Toke Høiland-Jørgensen and Andreas Kassler. Aggregation-Based Gossip for Certificate Transparency Logs. *The 13th International Conference on Emerging Security Information, Systems and Technologies (SECURWARE)*, Nice 2019.

   Relation to Included Paper: This work comprises our early work using the P4-NetFPGA platform and the OSNT traffic generator, which later guided our approach in Papers V, VI and VII.

Previous Papers

These are papers I have co-authored, which were published before starting my PhD studies and are listed for reference.

1. Peter Dely, **Jonathan Vestin** and Andreas Kassler. CloudMAC - Using OpenFlow to Process 802.11 MAC Frames in the Cloud. *PIK - Praxis der Informationsverarbeitung und Kommunikation, Volume 36, Issue 1.*

2. Peter Dely, **Jonathan Vestin**, Andreas Kassler, Nico Bayer, Hans Einsiedler and Christoph Peylo. CloudMAC - An OpenFlow based architecture for 802.11 MAC layer processing in the cloud. *Globecom Workshops*, Anaheim 2012.

3. **Jonathan Vestin**, Peter Dely, Andreas Kassler, Nico Bayer, Hans Einsiedler and Christoph Peylo. CloudMAC - Towards software defined WLANs. *Proceedings of the Annual International Conference on Mobile Computing and Networking (MOBICOM)*, Istanbul 2012. (Poster Paper)

Introductory Summary

1 Introduction

Computer networks have over time become a core component of the IT infrastructure in organizations and society as a whole. Today, most networks are typically deployed using closed, proprietary, and inflexible hardware employing rather static configurations. However, to meet rising customer demands, future networks must support new use-cases such as Industry 4.0, Internet of Things (IoT), and wearable IT. One important concept for these future use-cases is the idea of network convergence, essentially allowing a single network architecture to support any traffic type, from high-bandwidth video streaming to low-latency industrial automation. As network resources are limited, such a network must be highly flexible, with the ability to quickly provision the required network resources for each particular traffic flow. One prominent technology, that can provide such flexibility and enable network convergence, is Software Defined Networking (SDN) [1], which by opening up and logically centralizing the control plane, allows for programmatic control of the network infrastructure. While SDN enables control plane flexibility, the introduction of Data Plane Programming (DPP) [2], along with compiler and language support, lifts the flexibility of networking devices to a new level. Additionally, the concept of Network Function Virtualization (NFV) [3] is another strong enabler of network flexibility, allowing rapid deployment and precise orchestration of network functions.

Although SDN, DPP, and NFV increase network flexibility, there are still open challenges that must be solved before widespread adoption of these technologies can take place. Resiliency is one of these challenges, as use-cases like industrial automation, autonomous driving, and remote medical surgery have near-zero tolerance for network failures [4, 5]. Today, highly resilient networks often rely on closed, proprietary equipment with low interoperability, incompatible with the flexibility demand of future applications. Any network architecture used as a solution for introducing flexibility for such use-cases must have a mechanism for high resiliency. SDN provides several resiliency mechanisms. First, the logically centralized view of the control plane allows detecting link failures and optimally rerouting traffic flows. Also, SDN-enabled switches can perform rapid local link recovery, immediately rerouting traffic to a backup path, should the primary forwarding path go down. Finally, SDN has some preliminary features for switch queue configuration, which could be used to flexibly prioritize critical traffic. Together, these features may be the key to demonstrating the viability of SDN for flexible and highly-resilient networks.

Another aspect that must be considered is scalable network monitoring. Accurately providing appropriate network resources for each service flow, without overprovisioning, requires precise and up-to-date information of the network state [6]. Traditional network monitoring systems do not provide the highly-configurable per-service-flow information needed for future networks, but rather rely on inflexible and proprietary systems. With the introduction of DPP through SDN came the idea of In-band Network Telemetry (INT) [7],

which allows switches to append telemetry information to data plane packets. INT provides a basis for scalable, accurate and highly configurable network monitoring, and if good methods of telemetry insertion and collection are developed, it can prove viable even in large scale data center networks.

Finally, the idea of moving some operations, typically performed by the network control plane or by end-hosts, to the network fabric itself, is a powerful method to improve scalability and reduce latency of distributed applications. However, the inflexible design of traditional networking devices makes this difficult, as equipment tend to be closed and fixed in functionality. The introduction of DPP opens up the data plane for programming, which promotes an increase in network fabric intelligence. However, as programmable network devices can be limited in both expressiveness and performance, a good balance between in-network and end-host processing must be found.

In this thesis, we focus on how SDN and DPP can be used to improve the flexibility, resiliency, monitoring, and control for future networks. First, we will look at how SDN can improve the resiliency of Split-MAC Wireless Local Area Network (WLAN) networks, which are WLAN networks that logically centralizes association state. This is done by designing QoS extensions, making switch queue management strategies configurable from the control plane. Second, we use SDN-enabled local link recovery and packet duplication to improve the resiliency of both industrial automation and 5th Generation (5G) mmWave small cell backhaul networks, investigating the tradeoff between link recovery speed and traffic overhead. Third, we research how in-network telemetry event processing can be used to increase the scalability of network monitoring, through optimal configuration of telemetry probes and highly customizable event filtering in the data plane. Finally, we design and evaluate an in-network complex event detection system for industrial networks, which allows rapid reaction to sensor events, reducing latency, while also providing the resiliency required for distributed industrial control applications.

The rest of the thesis is structured as follows. Section 2 introduces the reader to SDN and the ideas of network resiliency, monitoring, and control, which serves as background material for the main contributions of the thesis. Section 3 describes the research objectives, while Section 4 briefly describes the scientific methods used. Section 5 presents the main contributions of this thesis work. Section 6 summarizes the appended papers. Finally, Section 7 provides concluding remarks and discusses future work.

2 Background and Related Work

This section will introduce the background of the thesis. Section 2.1 explains Software Defined Networking and how it can be used to provide network resiliency. Section 2.2 describes Data Plane Programming, along with the P4 language. Section 2.3 provides the background behind and the functionality of various resiliency improving technologies. Section 2.4 describes in-network control, stream processors and complex event processing. Finally, Section 2.5

explains in-network monitoring and discusses current approaches to event collection and filtering.

2.1 Software Defined Networking

Computer networks are separated into three logical planes: the *data plane*, the *control plane*, and the *management plane*. The *data plane* (also known as the forwarding plane), comprises software or hardware components of the network forwarding device, which performs the actual packet processing and forwarding. It typically consists of simple forwarding logic, often through the use of a forwarding table, and is most often implemented through high-speed dedicated hardware. The forwarding plane behavior is managed by the *control plane*.

The control plane takes care of higher-level functionality, with the main responsibility of populating the forwarding tables of the data plane, in order to control network state. Notably, the control plane decides where to send traffic through higher level decision logic. This could be a simple MAC learning table in a network switch, or be the communication with other network devices through routing protocols (such as the Border Gateway Protocol [BGP] [8] and the Open Shortest Path First [OSPF] [9]) in a router. As the control plane is typically implemented in software, that runs on a general-purpose CPU, packet processing takes significantly more time in the control plane compared to the data plane.

The control plane is in turn configured and maintained by the *management plane*. It includes protocols used for network device configuration, such as Telnet [10], Simple Network Management Protocol (SNMP) [11], Secure Shell (SSH) [12], and Netconf [13]. Network Management System (NMS) typically maintains and orchestrates the management plane in large network deployments, both monitoring device status and ensuring appropriate device configuration [14]. Due to early design considerations of the Internet, the control and data planes are traditionally tightly coupled, and the configuration is distributed [15].

To cope with the proprietary nature of computer networking devices, a new network architecture called SDN [1] was introduced. SDN separates the control and data plane of a traditional network device. The control plane is then implemented through one or more logically centralized controllers (also called Network Operating Systems [NOSs]). Each of these SDN controllers can manage multiple network forwarding devices, through a south-bound API (e.g., OpenFlow). Higher level logic can be implemented through applications on top of SDN controllers, which communicate through a north bound API (e.g., REST, JSON). Finally, SDN controllers can also coordinate network-control activities by establishing communication channels between each other using an east-west API, mainly for scalability and resiliency. These relationships can be seen in Figure 1, which shows a basic overview of a typical SDN architecture.

Much like the control plane in traditional networking, the SDN controllers manage the forwarding logic of the data plane, along with other control plane

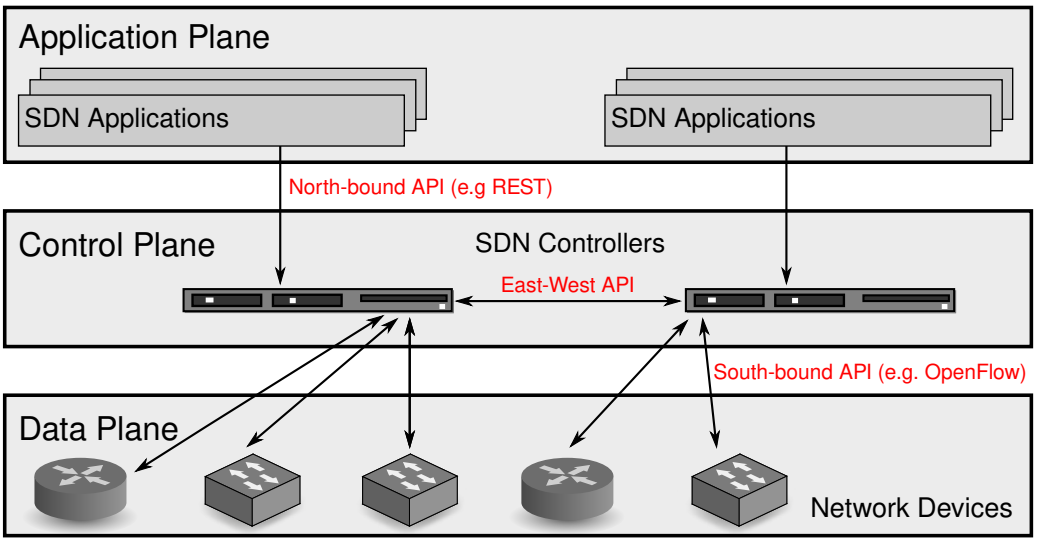

Figure 1: Overview of a typical SDN architecture.

functions, such as path calculation and traffic control configuration. This greatly simplifies the architecture (or design) of the data plane, as the control plane functions can be relocated to a separate, more powerful device. It also increases the flexibility of the control plane, which can obtain a centralized view of the network and, based on this view, calculate more optimal routing strategies and carry out more efficient traffic engineering. Furthermore, the SDN controller can be implemented in a high-level language, allowing network device manufacturers to keep the low-level implementation details of their devices hidden from competitors, while at the same time facilitating for network administrators and researchers to perform experiments, and to introduce new protocols and architectures using SDN-enabled devices. There are several SDN controllers in wide use today, e.g., Open Daylight [16], Ryu [17] and ONOS [18].

SDN applications are programs which perform higher-level network decisions and communicate the desired network behavior to the SDN controllers through their north-bound API. These applications range in responsibilities, including configuration and management, monitoring, troubleshooting, security, and accounting. One of the major responsibilities of these applications is SDN orchestration, which involves managing multiple controllers, ensuring Service Level Agreement (SLA) adherence and providing network administrators with monitoring information [19].

OpenFlow [20], which is an open protocol for programming the flow tables in SDN-enabled networking devices, spearheaded the development of SDN. Five major versions of the OpenFlow specification have been published [21]. One of the most popular software switches is Open vSwitch [22], which is a production-grade, multilayer virtual switch, supporting all major OpenFlow versions. OpenFlow suffers from one major drawback, which is that the programmability only extends to the control plane. This means that the data plane protocols supported are inherently limited by the current OpenFlow

specification. Additionally, operations that require a rapid reaction to changes in network state may not be easily implementable. These include operations like stateful forwarding, i.e. allowing the data plane to store information from incoming packets and use this to augment forwarding decisions.

2.2 Data Plane Programming

The introduction of SDN has made great strides in opening up network forwarding devices, for both industry and academic use. However, there are still a number of design limitations preventing full control of the network behavior. One of these limitations is the fixed set of header fields[1], which OpenFlow-enabled switches can process. Over time, more header fields have been added to the OpenFlow specification, meaning older equipment might support a smaller subset of headers compared to newer equipment.

To combat these challenges, several solutions were introduced to improve the programmability of network equipment. These solutions can be divided into two groups, the first one being *host-based* solutions. Host-based solutions aimed to improve the performance of end-host network processing. These are typically used in cloud deployments on hypervisors to serve in conjunction with SDN-enabled virtual switches. One widely used host-based solution is the Intel Data Plane Programming Kit (DPDK) [24], which is a set of libraries to accelerate user-space packet processing. Another important host-based solution was the eXpress Data Path (XDP) [25], extended from Extended Berkley Packet Filters (eBPF) [26]. XDP enables installation of small network programs in the Linux kernel, which are invoked each time a network packet is received before the kernel performs any other packet processing. From XDP, `AF_XDP` [27] was introduced, which in the same way as DPDK accelerates user-space packet processing.

The other group of solutions is *switch-based* solutions. This group of solutions aims to improve the processing flexibility of network switches. One such solution is OpenState [28], which extended OpenFlow with stateful operations, essentially allowing it to update its forwarding behavior without requiring interaction with the control plane. The software switch Open vSwitch [22] (e.g., used in the Pica8 [29] hardware switch), also made several efforts to increase the programmability of the network switch, including executing eBPF [26] programs on incoming packets [30]. Another switch-based solution is SnabbSwitch [31], which is a virtual switch, programmable in the high-level programming language Lua.

One of the major challenges with these solutions is how to provide a good balance between speed and programmability. Some operations, which may be very useful, may also be very costly for a packet forwarding device expected to run at line rate. One of the solutions, which attempt to combine both expressiveness and performance, is Programming Protocol-Independent Packet Processors (P4) [2]. This solution was developed from the abstract switch table model, Reconfigurable Match Tables (RMT) [32]. The P4 language is

[1] There are 12 fixed header fields in OpenFlow 1.0 [23]

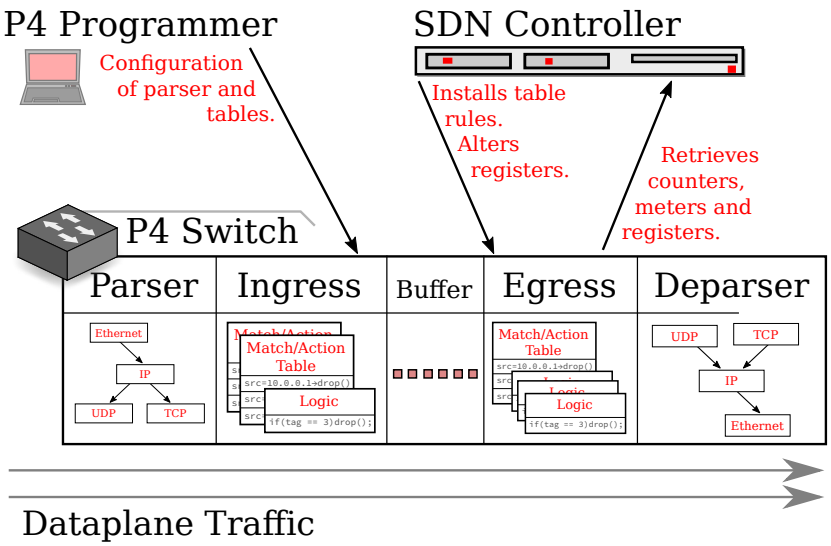

Figure 2: One of the possible data path abstractions provided by P4. This particular data path abstraction is used in the BMv2 software switch [35].

used to define the data plane behavior of a network device. There are currently two versions of P4 in use, the older version P4_14 [33] and the newer version P4_16 [34]. As shown in Figure 2, P4 programs are typically divided into four principal parts: a *parser*, an *ingress* pipeline, an *egress* pipeline, and a *deparser*, although which parts that are included in the programmable part of the processing pipeline and the order of these parts may differ among manufacturers. The *parser* is structured as a state machine, where each node in the state machine represents a parser state. Incoming packets are processed from the starting state, and parsing is complete when the current state concludes in either an *accept* or a *reject*. The parser can be recursive, in order to parse header stacks. However, each P4 target typically has a limitation on recursion depth.

The *ingress* pipeline consists of a *control* block, syntactically quite similar to the C programming language: `if`-clauses can be used for conditional execution, local variables can be defined and modified, and packet headers and persistent switch memory (registers, meters, and counters) can also be modified. Another component which P4 control blocks have access to is match/action tables. By using these control blocks, P4 switches expose their forwarding behavior to an SDN controller. A P4 table has, like an OpenFlow table, a defined match-action tuple, which consists of one or more header fields and a set of actions. After ingress processing is complete, the packet is typically moved to an internal packet buffer. At the time the packet is removed from the buffer, *egress* processing is carried out on it. The egress pipeline functions similarly to the ingress pipeline, consisting of control blocks and forwarding tables. Finally, the packet reaches the *deparser*, which re-assembles the packet and sends it to the specified output port. In-between egress processing and deparsing, some switch targets also have a re-calculate checksum step.

Although the ingress and egress processing look very much alike regular C code, there are several inherent limitations imposed on this programming

model. Particularly, since packets most of the time have to be processed at line rate, the ingress and egress processing have to be timely. To this end, floating-point arithmetic, integer multiplication, and integer division are unavailable. Integer multiplication can be performed, provided the multiplication entails multiplying two constants, or a variable and a constant, as the multiplication can be transformed into a series of additions. Also, loops are not available, as their execution time may be unbounded. Instead, loop unrolling can be used in place of loops. In addition to these limitations, each target may have additional limitations, depending on how the software or hardware is designed. Typical limitations include program complexity, memory size, header size, and parsing tree size.

Pure P4 is a standardized language, and a program written in P4 should be executable on any target platform. However, P4 also supports extensions which makes it possible for manufacturers to include purpose-built functionality that goes beyond the P4 standard. These external primitives, or `externs`, are specialized, target-dependent functions. Examples of typical P4 extensions include hashing functions, encryption/decryption, floating-point or other mathematical operations, and other functionality excluded from the P4 standard to keep it simple and compatible with a wide range of devices. Many platforms also allow developers to implement custom external functions, typically in a specific, target-dependent language. These languages include Micro C, for the Netronome Agilio platform family [36], and Verilog, for the P4-NetFPGA [37] platform family.

Today, there is a wide range of P4 compilers and target platforms available. For example, the Behavioral Model (BMv2) [35] is a software switch acting as a reference implementation. While not a production-grade software switch, it is still useful for developing, debugging and experimenting. The behavioral model is typically used with the `p4c` [38] compiler, which is the reference compiler for the P4 language. The Netronome Agilio [39] is a hardware platform, which has support for the P4 language, through the use of the Netronome datapath programming tools. Another hardware platform is Barefoot Networks Tofino [40], which is based on the Protocol Independent Switch Architecture (PISA) [41]. There are also a set of compilers, which target additional platforms, such as P4-NetFPGA [37], which compiles P4 code to the NetFPGA [42] platform. Another compiler is `p4-xdp` [43], which uses `p4c` as a base, compiling P4 code to an XDP program. Finally, the T_4P_4S [44] compiler is able to compile P4 code to multiple targets (Intel DPDK, among others), through the use of a network hardware abstraction layer.

Many powerful applications have been developed using the P4 language. For example, the authors in [45] implement and evaluate a local link repair primitive through the use of P4, with functionality similar to fast failover. Another example is P4CEP [46], which presents an in-network approach to Complex Event Processing (CEP), using the P4 language. They compare their approach to other CEP processing platforms, such as Apache Flink. Another example is in-network telemetry [7], which is a P4 framework designed to facilitate network monitoring through the data plane, by having telemetry

information piggyback onto regular data plane traffic.

2.3 SDN-Enabled Resiliency in Computer Networks

Resiliency is the ability of a network to maintain a certain level of service, when exposed to external factors, that disrupt network functionality. Examples of such external factors may include equipment failures, power outages, signal fading, software or human errors, denial of service, and malicious service disruption. Network resiliency, as such, involves quickly identifying and recovering from such failures [47, 48]. While the area of network resiliency is broad, this thesis mainly focuses on network link resiliency, i.e., how networks can cope with link or device failures.

In traditional networks, link resiliency technologies can be divided into two groups: *alternate path* protocols and *parallel operation* protocols [49]. *Alternate path* protocols provide one or more alternate traffic paths, that are used if the primary network path fails. Examples of alternate path protocols include Spanning Tree Protocol (STP) [50], Rapid Spanning Tree Protocol (RSTP) [51], and Media Redundancy Protocol (MRP) [52]. The other type of resiliency mechanism is *parallel operation*. In parallel operation, data is replicated over multiple disjoint paths, which makes link recovery more or less instantaneous. However, this requires concurrently running network devices, which can increase both installation and operational costs. Furthermore, these protocols often require specialized hardware and/or software support in the end nodes to detect and remove duplicates. Examples of parallel operation protocols include Parallel Redundancy Protocol (PRP) [53], IP Parallel Redundancy Protocol (iPRP) [54], and High-Availability Seamless Redundancy (HSR) [55].

Future network applications, such as Industry 4.0, IoT, autonomous driving, and remote medical surgery, impose new requirements in both network flexibility and resiliency. Traditionally, highly resilient networks are often powered by inflexible, proprietary equipment with low interoperability (such as HSR and PRP [55]-enabled proprietary Industrial Ethernet equipment). In other words, they are incompatible with the flexibility requirements of future networks. SDN is one approach which can provide the high flexibility required, while also sustaining high resilience.

In SDN, there are two main approaches to link recovery: *proactive* and *reactive* recovery [56, 57]. In the *reactive* approach, when a link fails, the SDN controller is informed, re-calculates new forwarding rules, and installs them in the device where the link failed. Other devices may also have their forwarding tables updated as required. As the controller has to receive information about the link failure, calculate new forwarding rules, and perform the rule installation, a reactive recovery of a link typically takes a relatively long time, in the order of 50 ms [58].

Compared with the reactive approaches, the *proactive* approaches reduce the link recovery time by pre-installing the available backup paths in the forwarding devices. The forwarding devices continuously monitor link liveness, and should the link go down, one of the live alternate backup links are used

instead. The downside of this approach is that the sizes of the flow tables are increased. Further, failures where there are no backup paths requires additional maintenance of local state, such as using Crankback forwarding [57], i.e., sending packets back until they reach a switch that can forward them on an alternative path or a state-based solution, such as OpenState [28]. Ideally, when SDN is used to provide network resiliency, a combination of proactive and reactive recovery approaches should be used. This makes it possible for the network to quickly recover from a link failure through a proactive recovery, and when the SDN-controller detects the broken link, it can recalculate optimal paths and install new rules reactively.

With the introduction of OpenFlow 1.1 [59], support for group tables was added. A group table is a group of output ports combined with a group type. Instead of forwarding an incoming packet to a specific port, the controller can forward it to a group table. The group table will then, depending on the group type, select one or more of its ports, and forward the packet via the selected port. Fast failover is a group type that sends packets to the first live port. If no ports are considered live, the packets are dropped. This is considered a type of *proactive* link recovery. Fast failover requires that the ports in a group table have some kind of liveness detection method configured. There are multiple methods for detecting the liveness of a link, such as relying on Ethernet Normal Link Pulses (NLPs) or using another protocol like Bidirectional Forwarding Detection (BFD) [60]. The popular software switch, Open vSwitch [22], implements fast failover and uses BFD for liveness detection. Recent improvements in Open vSwitch, allow for very low BFD message intervals. This enables Open vSwitch to detect and recover from link failures within (3.3 ± 0.8) ms [57].

2.3.1 SDN-Enabled Resiliency in Industrial Automation Networks

Industrial automation networks are networks used in industrial applications, such as Profinet [61] and Modbus [62]. These networks are typically closed and tightly integrated, making interoperability with other networks challenging. The application areas are very diverse and include manufacturing systems, energy and power systems, automotive communications, avionics, aerospace, and railway control systems [49]. What these systems have in common is the need for fast and reliable machine to machine communication [63, 64]. For example, avionics might require communication between heading sensors and navigation systems while a manufacturing plant might need synchronization between motor actuators and photosensitive sensors. With the introduction of Industry 4.0 [65] and IoT [66], these industrial automation systems will put even higher requirements on both timeliness and resiliency. As standard Ethernet was not designed with timeliness or resiliency in mind, it cannot provide deterministic packet delivery, and it lacks built-in resiliency mechanisms.

Time Sensitive Networking (TSN) [67] is a recently introduced improvement of Ethernet, which aims to make it more deterministic and suitable for industrial automation applications. TSN currently encompasses roughly 12

IEEE 802 standards [68], together aiming to provide deterministic networking, with a high quality of service, without compromising on interoperability. Of particular interest is IEEE P802.1CB Frame Replication and Elimination for Reliability [69], which seeks to guarantee latency during single points of failure. It does this through selectively duplicating packets at the sender and discarding duplicates at the receiver, providing parallel operation resiliency.

While TSN seeks to make Ethernet more time-sensitive and fault-tolerant, it increases the number of network parameters that need to be configured and maintained. Further, their configuration may depend on data derived from a global network view, such as global monitoring information, traffic patterns, and network topology. With an SDN solution, the controller can manage this configuration, and thus provide higher-layer service guarantees. There are many ways in which SDN can be utilized in an industrial setting [70, 71, 72, 73] and several of these ways use SDN to manage the parameters in TSN [74, 75, 76]. Resiliency in SDN-enabled TSN networks can be provided either from TSN, through IEEE P802.1CB, or from SDN, through the use of packet duplication combined with fast failover groups and BFD. The challenge of the SDN approach is duplicate identification and discarding at the destination, which has to be performed by the end-host, or in the network using programmable data planes.

Throughout this thesis, we will consider the idea of integrating SDN and NFV in an industrial automation network, essentially creating a converged industrial network able to support any type of traffic. Here an SDN orchestrator could, from the characteristics of each industrial application, configure the entire service chain. This could include TSN parameters, resiliency configuration, timing requirements, bandwidth reservation and queue management strategies, depending on the particular equipment. Such a network may be one step towards supporting the diverse traffic future industrial applications may produce, and provide a simple and consistent interface, where automation parameters could be directly and seamlessly translated to network configuration.

2.3.2 SDN-Enabled Resiliency in 5G mmWave Small Cell Backhaul Networks

With the emergence of the 5G mobile networks, high demands are placed on both throughput and latency, targeting 1 Gbps – 10 Gbps throughput and sub 1 ms latency [77]. The 4th Generation (4G) Long Term Evolution (LTE), which is the currently deployed solution for high data rate mobile networking, has seen a significant capital and academic investment in the development of new technologies to facilitate even higher throughput. These improvements include the deployment of Massive Multiple-Input and Multiple-Output (MIMO), the introduction of Co-ordinated Multipoint (CoMP), improved coding, and modulation schemes [78]. However, despite these improvements, traditional frequency bands encompass a small, heavily used spectrum, and cannot provide the bandwidth required for such high-speed transmissions.

One solution to cope with the high bandwidth requirements of 5G, is

to utilize parts of the Extremely High Frequency (EHF) spectrum, such as the mmWave bands (typically between 30 GHz and 300 GHz) [79, 80, 81, 82]. Using these frequencies provide very high potential bandwidth, up to 100 Gbps, and allow mobile stations to communicate with base stations at the high data rates required by future applications. These EHF bands have traditionally not been considered due to their short-range and non-line-of-sight coverage issues. In these frequency bands even rain and thick foliage may incur propagation losses. In order to deal with these propagation losses, technologies such as directional beamforming antennas are often deployed. These antenna arrays can circumvent some blockage [83]. However, the prevalence of blockage in urban environments may still cause intermittent outages and thus fluctuation in available bandwidth.

Mobile network cells are typically connected to the core network through fiber links, or point-to-point high-speed microwave wireless links. However, with the increasing density of mobile cells, this method of providing core network connectivity can be cost-prohibitive. As such, one strategy for reducing costs is using the mmWave bands also to provide backhaul connectivity to the core network. SDN [84, 85, 86] has been suggested as a method to coordinate these small cell backhaul networks. An additional advantage to using SDN for the backhaul network, is that SDN-enabled local link recovery could be utilized to recover from the rapid link quality fluctuations present in mmWave frequency bands.

2.3.3 SDN-Enabled Resiliency in Split-MAC WLAN Networks

Future network use-cases, such as autonomous driving, IoT, and Industry 4.0 require high network flexibility and resilience. In mobility scenarios, WLAN networks suffer from connectivity loss during handover between access points, which is incompatible with the near-zero tolerance for failures of future network applications. One method to solve this is to provide seamless handover through a Split-MAC approach [87]. In a Split-MAC network, the WLAN station association state is separated from the access point and relocated to a logically centralized controller. There are multiple approaches of how to realize a Split-MAC. One approach is Odin [88], where Light Virtual Access Points (LVAPs) maintain the association state of the WLAN stations, and an SDN controller handles the handover between different LVAPs. Other approaches include CAPWAP [89] and its predecessor LWAPP [90]. CAPWAP uses a centralized controller to discover WLAN APs, which also provides authentication and authorization services to them. In this approach, some WLAN management frames are generated by the CAPWAP controller, while other frames are generated by the APs.

CloudMAC [91] is an additional Split-MAC approach, which functions similar to Odin, however, it moves all Access Points (AP) functions to a Virtual Access Point (VAP), which is deployed in a co-located cloud, and transforms the physical APs into lightweight Wireless Termination Points (WTPs). These WTPs simply tunnel traffic from one or more VAPs, while also responding

to time-critical WLAN acknowledgements. An SDN controller manages the connection between WLAN stations, WTPs, and VAPs, and performs seamless handovers between WTPs, by rewriting flow rules to redirect traffic towards the new WTP. Indeed, as the association state is maintained by the VAP in the cloud, the mobile station is always associated and the flow routing is updated according to the mobility of the station. A difficulty with these approaches is the timing of management frames. When WLAN stations connect to an access point, the access point expects timely responses to management and control frames [92, 93]. For example, the response time for a WLAN acknowledgment is as short as 10 µs using 802.11g [94]. As management and control frames are processed on a separate host, WLAN connection stability can be significantly reduced by congestion, and connectivity may be lost.

One option to increase the connection stability of such networks is employing queue management strategies in order to prioritize CloudMAC management frames, such as Hierarchical Token Bucket (HTB) [95], Stochastic Fair Queueing (SFQ) [96] or, more recent Active Queue Management (AQM) strategies such as Controlled Delay (CoDel) and FlowQueue-Codel (FQ-CoDel) [97]. As most Split-MAC networks maintain the connection between VAPs and stations, through the use of an SDN controller, queue management strategies must be configured through this controller. Several approaches that allow SDN controllers to configure network queue management have been proposed, such as QoSFlow [98] and FlowQoS [99]. One of the weaknesses with these approaches is that the extensions are unavailable on production-grade switches. Currently, only Open vSwitch allows configuration of queue management through the platform-agnostic `ovsdb` [100] protocol, although it only supports the HTB and Hierarchical Fair Service Curve (HSFC) queue management strategy, and lacks support for modern AQM strategies.

2.4 SDN-Enabled In-Network Caching and Control

Future application areas, such as 5G mobile networks, autonomous driving, and smart cities will significantly increase the processing capacity demand on computer networks. To provide a scalable solution to this development, a promising idea is to partially distribute the application logic, typically performed by end-hosts, to the network fabric [101, 102]. This is beneficial, as certain application logic is easily implemented with low complexity input-output processing techniques. This enables them to be deployed directly in-network, reducing the requirement for dedicated end-hosts.

Two approaches, which provide such functionality, are in-network caches and in-network Complex Event Processing (CEP). The idea behind caching comes from Online Service Providers (OSPs), such as web search engines, social networks, and web stores, where a single web request could generate hundreds or thousands of storage accesses [103]. To make such services possible, providers typically rely on in-memory key-value stores, which cache frequently accessed pages [104]. One example of in-network caching enabled

through the use of data plane programming is NetCache [102], which provides an in-network, load-balancing, medium-hit-ratio cache, with a prototype implemented for the Barefoot Tofino platform supporting up to 2 billion user queries per second. Another approach is NICached [105], which instead uses XDP, and increases cache reply speed for typical end-hosts by performing the lookup at the earliest point, when the packet reaches the device.

CEP, on the other hand, involves inferring more complex events from a set of smaller, less complex events. For example, a complex event might be a fire outbreak in a manufacturing plant, which could be inferred from, e.g., a smoke detector and a temperature meter. Here, a sudden rise in temperature together with smoke being detected by the smoke detectors could indicate an outbreak of fire, and thus should trigger a fire alarm [106]. Complex events can, as such, involve multiple sensors reporting specific values over a range of time, where in order to detect the event, historical events must be stored. Industrial automation is one area that involves the detection of complex events, and could benefit from moving its event detection to the network fabric. For example, it could result in faster reaction time to critical events, as the event detection is placed in-network, closer to the sensors and actuators. Additionally, as the sensor-actuator loop is shorter, there are fewer points of failure in the network, increasing overall resiliency. Finally, compared to conventional stream processors, like Apache Flink [107], in-network CEP is faster, with more deterministic latency, which is an important aspect for industrial automation networks.

P4CEP [106, 46] is one method for processing complex events in the data plane through the use of P4. In P4CEP, events are expressed in a high-level CEP language, that are compiled to P4 code, which in turn is compiled to a switch program. The authors evaluate their approach using the Netronome Agilio NFP4000 platform, showing a significant improvement in performance compared to Apache Flink and the BMv2 software switch. A similar approach is Packet Subscriptions [108], which provides content-based routing through the use of CEP-like expressions, such as "`stock = MSFT and price < 60 and shares > 100`", where the values for `stock`, `shares`, and `price` are provided from the packet payload. The authors use a two-stage approach, where they compile a specification into a P4 program. When the resulting switch program is compiled and installed in the switch, they generate and install a set of table entries that specify the exact forwarding rules. The Packet Subscriptions approach has been evaluated on a Barefoot Tofino switch and offered in this evaluation a packet processing time of less than 20 µs. One of the disadvantages with these approaches is that when the CEP expression is compiled and installed in the switch, the control plane is limited in altering its behavior. This limits the flexibility of these approaches because the controller cannot update the control logic of a networking device without installing a new P4 program.

2.5 SDN-Enabled In-Network Monitoring

Providing network operators with accurate and up to date knowledge of network status is crucial in modern network deployments, to provide high-quality user experience and/or SLA conformance. However, as future networks must provide high flexibility, and at the same time scale with the throughput demand from next generation applications, future network monitoring approaches must be both flexible and scale with the size of the network. These requirements became even more important with the introduction of NFV [3] and Service Function Chaining (SFC) [109, 110], as each service flow and network function must be monitored to ensure a correct and SLA conformant handling of packets in the network. With the introduction of SDN and DPP came the idea of In-band Network Telemetry (INT) [111], which is a method for providing network monitoring through the data plane. The processing required from the data plane to support INT is comparatively small, allowing per-packet monitoring to be performed at line rate with minimal impact on regular traffic. Additionally, using DPP to provide INT enables network operators to individually customize monitoring characteristics for each service flow, enabling high flexibility.

In an INT-enabled network, each node along a network path adds telemetry information to incoming packets, which at the end of the path is removed and sent to a network monitoring platform. These telemetry items may include ingress timestamps, queue occupancy information, switch identifiers, per-packet selected ingress/egress port, and various other metadata. INT-enabled networks typically identify three different roles, which each forwarding device can take: *source*, *in-transit*, and *sink*. The *source* nodes insert the `telemetry` header, along with an instruction bitmask. The *in-transit* nodes identify incoming packets that contain a `telemetry` header, look up the instruction bitmask, and add INT `metadata` headers according to the bitmask. Finally, the *sink* removes the `telemetry` and `metadata` headers, and forward the collected information to an event monitoring system. This event monitoring system is typically a stream processor, which may act on the incoming data, or store it in a database for offline analysis.

Many INT-based solutions to network monitoring have been proposed. One of the solutions is INTCollector [112], which improves the telemetry processing capacity of end-hosts, by offloading processing to a fast path utilizing XDP. It has support for both interval-based and threshold-based event filtering. Another solution is Sonata [6], which is a framework that enables complex, in-network telemetry event detection. In particular, it splits event processing between user-space stream processors and programmable data planes. This approach enables the use of powerful stream processing operators such as *filter*, *map* and *reduce*. The authors in [113] present an approach using in-network telemetry as an enabler for Knowledge Driven Networks (KDNs). They demonstrate the INT approach for KDN in several use cases, including traffic engineering and anomaly detection. Selective In-Network Telemetry (sINT) [114] is an approach to telemetry collection, where the collection rate of packets can be adjusted according to the frequency of important changes

in the network information. INT sinks use data provided by the INT sources to determine this insertion ratio. Keysight [19], which improves upon Netsight [115] uses in-network telemetry to troubleshoot switch data plane code, through the use of packet postcards, that contains the forwarding path and state of the packet itself. To detect errors, the postcards are compared to acceptable network behavior. One key challenge of using INT is finding a good balance between the accuracy of the captured network state and the overhead of network telemetry processing. While previous approaches such as INTCollector and sINT improve processing speed by event filtering, the monitoring accuracy is reduced as important events may be filtered out.

3 Research Questions

For future converged networks to support next generation use-cases, such as Industry 4.0, IoT, and wearable IT, they must be flexible, dynamic, and scalable, with the capacity to support multiple divergent traffic types simultaneously. One method which can enable such flexibility is SDN, and by extension DPP, which both attempt to open up traditionally closed network equipment for programming. The objective of this thesis is to explore how SDN and DPP can improve the resiliency, monitoring and control capabilities of next generation networks. To achieve this research objective, we seek to answer the following research questions:

1. *Can SDN be used to improve resiliency in computer networks?*

   Computer networks that require high availability could benefit from the flexibility and reliability which SDN provides. The centralized controller can improve reliability through queue management configuration, reducing network congestion, and through active link recovery, allowing switches to rapidly repair link failures. While these two approaches can improve the reliability in SDN-enabled networks, their use in real-world networks remains partially unexplored. We set out to investigate these SDN-resiliency features in three divergent use-cases: Split-MAC WLANs, industrial automation and 5G mmWave small cell backhaul networks. As each of these use-cases has widely varying characteristics, our goal is to find both an effective approach for how to utilize the resiliency provided by SDN and evaluate that approach.

2. *Can DPP and in-network telemetry be leveraged to improve the scalability and flexibility of network monitoring?*

   The introduction of DPP, and in particular the P4 language, promoted the idea of in-network telemetry as a monitoring solution for future networks. By inserting telemetry data in the network fabric, precise, scalable, and flexible network monitoring is possible. Although a promising approach, the performance impact remain largely unexplored. Additionally, telemetry collection is traditionally performed through the use of end-host stream processors, which do not scale well in larger network

deployments. Our objective is to research and develop solutions that improve the scalability of telemetry collection, which enable the use of in-network telemetry in large data center networks.

3. *Can DPP be used to increase the switch intelligence, enabling processing of complex events in the data plane?*

   Through the introduction of DPP, the data plane is opened up and network equipment become significantly more programmable. This allows moving complex event processing, typically performed in the control plane or by end-hosts, to the data plane. For example, parts of the distributed automation logic in an industrial automation setup, network monitoring or general stream processing, could be moved to the data plane. However, data plane programs often have restrictions in expressiveness and must execute in bounded-time, processing packets at line rate. We aim to create a highly flexible and high-performance framework for complex event processing in the data plane. To demonstrate its viability, we will use an industrial automation event processing use-case to evaluate the tradeoff between performance and flexibility.

4 Research Methods

Computer Science can essentially be divided into three different categories: hardware, which examines the physical structure of computational devices, software which examines the software that is executed on the hardware, and theoretical computer science, which examines the capabilities of computational methods through analyzing and designing algorithms and data structures [116]. With the aim of answering the research questions formulated in Section 3, we will examine a combination of these aspects in a computer networking context, including the performance and limitations of networking hardware, the implementation and design of networking software and some consideration for algorithms related to networking.

As the main method for answering our research questions, we will utilize the scientific method [117]. The scientific method is an approach for acquiring knowledge, following an iterative process, that involves careful observation and rigorous experimentation. The use of the scientific method, generally go through a series of stages, each depending on the previous one. How these stages relate to each other can be seen in Figure 3. To demonstrate the stages of the scientific method, we use our approach for Paper I as an example:

Hypothesize Formulate a hypothesis that can explain an observation. The resulting hypothesis must be falsifiable, otherwise it cannot be tested. For example, in the early stages of Paper I, we crafted a hypothesis, which stated that the cross-traffic in CloudMAC negatively impacts connection establishment.

Design Create an experiment that can test your hypothesis. In Paper I, the

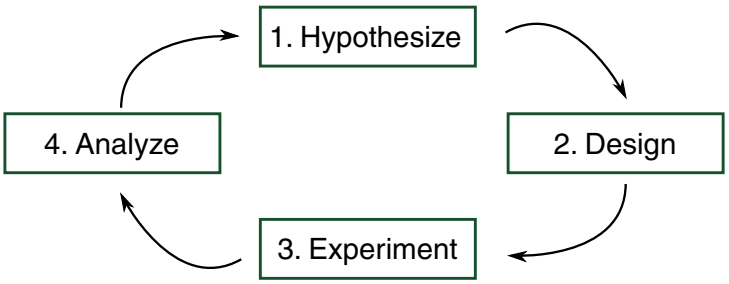

Figure 3: The iterative process of the scientific method.

hypothesis was tested by simulating cross-traffic and measuring the connection success rate.

Experiment Perform the experiment. Repeating the experiment multiple times help improve our confidence in the result. The number of repetitions depends on the variability of the experiment, and the requirements of the analysis.

Analysis Analyze the results of the experiment(s). If they do not match your hypothesis, revise or reformulate your hypothesis or formulate a new one. In Paper I, the results showed that the connection success rate was impacted. We observed that the management frame delay, together with the employed WLAN hardware, to a large extent determined whether a connection was successfully established or not. This resulted in the formulation of a new hypothesis, that using traffic control in the CloudMAC network would improve the success rate.

The experiment step of the scientific method is of particular interest and is in computer networking typically conducted in three different ways: through *simulation*, *testbed experiments*, and *emulation*. Simulations often use software which contains algorithmic and mathematical models, that make simplifying assumptions on how network elements behave. Simulated computer networks generally provide high accuracy, as long as the model used to simulate the network is realistic. However, some parts of real computer networking can be extremely difficult to simulate precisely, e.g., wireless networks. Testbed experiments, on the other hand, uses real-world hardware, software, and configurations, which can provide very accurate results. However, testbed experiments with real traffic many times comes at the cost of obtaining equipment, which may require significant capital investment. For certain experiments, there may not even be suitable equipment available, as it has not yet been created. Another weakness of testbed experiments is the increased amount of variability, which makes repetition more difficult. For example, performing a wireless experiment may have interference caused by nearby objects or people. It is also difficult to reproduce an already published testbed experiment, which involves obtaining the right equipment, the correct setup, and the correct configuration. Finally, we have emulation, which is a hybrid approach between the simulation and testbed experiments. In emulation, physical hardware is

Paper	Emulation	Testbed Experiment
Paper I		Measuring QoS impact on CloudMAC
Paper II	Evaluation of resiliency in industrial automation networks using the CORE emulator	
Paper III	Evaluation of fast failover in mmWave networks using the CORE emulator	
Paper IV	FastReact evaluation using BMv2 software switch through the CORE emulator	
Paper V		Evaluation of INT header insertion overhead using P4-NetFPGA
Paper VI		Evaluation of INT event filter and Kafka message production on physical hardware
Paper VII		FastReact evaluation using NFP4000 SmartNIC

Figure 4: Per-paper table of experimental methods used.

used for some parts of the network, such as the host machines, while other parts of the network are simulated.

While all of these methods have their respective advantages and disadvantages, it is important to note that to validate a hypothesis, a combination of these approaches should be used. In this thesis, both emulation and testbed experiments are used to validate our hypotheses. Table 4 shows a table, summarizing the experimental methods in each paper included in this thesis.

Emulation

In both Paper II and Paper III, an emulation approach was used, utilizing the CORE emulator [118]. This emulator uses Linux containers and network namespaces to create a large number of virtual machines running on the same physical computer. These virtual machines are connected using virtual bridges, which can be configured using the traffic control (`tc`) system in Linux. The emulator can emulate link characteristics such as bandwidth, delay, jitter, and packet loss, while still using real traffic and real applications. In both experiments (Paper II and III), we extended the CORE emulator to support Open vSwitch [22], a popular OpenFlow software switch. We chose to emulate the network setup, because performing physical experiments with large networks or new technologies can be quite challenging, and emulation gives us a near-realistic environment while allowing us to construct a cost- and time-effective testbed. In Paper II, as we were interested in the performance of local-link repair using fast failover in industrial automation networks, we mainly focused on the packet loss rate and latency. However in Paper III, as we investigated the feasibility of using fast failover in future mmWave 5G networks, we included more metrics, such as: throughput, delay, BFD failover count, and overhead.

We utilized our experience with emulation and the CORE network emulator, adding support for the BMv2 P4 software switch. This extended emulator was used to evaluate FastReact (Paper IV), our idea for how P4 can be used to partially move end-host processing to the data plane. Through an evaluation

with the CORE emulator, we compared the latency obtained with our switch program with the latency obtained in a regular industrial automation network setup. Further, we evaluated the recovery time during a link failure for our P4 implementation of fast failover.

Testbed Experiments

In Paper I, we performed the evaluation using a testbed approach with desktop PCs, WLAN access points and off-the-shelf switches. To evaluate the performance of the network, we studied the latency and throughput, response time, and the WLAN connection success rate. This was tested with and without cross-traffic, using various traffic control algorithms. We chose this validation method because CloudMAC is designed to work with off-the-shelf WLAN stations, and should continue to do so even after our improvements.

In Paper VII, an experiment testbed was used for our P4 programs. The testbed was built using both Netronome NFP4000 [36] SmartNICs and the T_4P_4S compiler [44]. The physical hardware, while being more challenging to use, gave us significant insights into how FastReact behaves in a real network. We looked in detail at the effects of performing complex event processing in the data plane on latency and throughput. For Paper VI, we used the Netronome NFP4000 hardware platform for our P4 switch program and combined this with `AF_XDP` to increase the INT report collection capacity of a network. We primarily measured throughput and, based on the measured throughput, we extrapolated processing capacity from it. Finally in Paper V, we used a different P4 hardware platform, called P4-NetFPGA [37] to explore the impact of pushing INT headers on packet processing time.

5 Main Contributions

This section summarizes the main contributions of this thesis and how the contributions relate to the research questions formulated in Section 3.

> **Can SDN be used to improve resiliency in computer networks?**

Our first contribution towards answering this research question was improving the connection stability, and thus resiliency, of Split-MAC WLAN networks. These networks are commonly SDN-enabled, using the controller to manage handovers between access points. We extended Open vSwitch with classless qdiscs[2], namely SFQ, CoDel and, FQ-CoDel, measuring their impact on connection stability in CloudMAC, a Split-MAC network. From the results, we can conclude that the connection stability is significantly increased, even in networks where the wired part is heavily congested, through the use of either SFQ, HTB or, FQ-CoDel as a queue management strategy to prioritize WLAN management frames. Through the implementation, evaluation, and

[2] Available in mainline Open vSwitch as commit 677d915

results, we contribute a strong argument for using SDN to provide queue management in Split-MAC networks, as a method for improving connection stability. These contributions were presented in Paper I. Further extensions, including Quality of Service (QoS) improvements, using IEEE 802.11ae access categories, Grafana [119] and InfluxDB [120] integration in the controller, were presented in [121].

Our second contribution towards this research objective was investigating the usage of SDN-enabled *proactive* link recovery in both SDN-based industrial automation and 5G mmWave small cell backhaul networks. In the case of SDN-based industrial networks, our design allows for both parallel operation (duplicating traffic) and alternate path (fast failover) approaches. These two approaches were both evaluated separately, and in combination, showing an almost-zero packet loss, even under bad network conditions. This contribution was presented in Paper II. We also evaluated similar techniques in an emulated 5G mmWave small cell backhaul network, utilizing a mathematical model to simulate the periodically changing network conditions prevalent in such high-frequency networks. Using SDN-enabled link recovery, with alternate paths consisting of both other mmWave nodes, and a direct LTE link to the eNodeB, our approach was able to cope with rapid fluctuations in link stability. It also significantly reduced packet loss and increased available throughput. The idea was presented as a poster in [122] and expanded through the use of a more accurate model and a LTE assist mode in [123], and further expanded including TCP and UDP throughput, overhead costs and failover count metrics in Paper III. In combination, these contributions demonstrate an approach where both SDN-enabled parallel operation and alternate path methods can improve the resiliency of SDN-based computer networks. Due to the SDN based architecture, resiliency parameters are configurable on a per-flow basis, and can be tailored to each traffic type.

Can DPP and in-network telemetry be leveraged to improve the scalability and flexibility of network monitoring?

To answer the second research question, we evaluated the impact of the INT header and data insertion, using the P4-NetFPGA platform. Through these results, we contributed accurate measurements on the per-item cost of INT telemetry insertion, both in INT source and transit nodes. These measurements were used in Paper V in order to optimize the active probing of network slices. Additionally, we investigated how to improve the telemetry collection capacity of INT networks, through intelligent in-network event filtering and high-speed telemetry collection. We contributed both a method and an implementation of highly flexible in-network INT event filtering. Additionally, we designed and implemented a high-speed telemetry collection program, utilizing `AF_XDP`. We evaluated both of these contributions together, showing significant improvement in INT collection capacity. We believe this finding may strengthen the idea of INT as a scalable alternative for network monitoring in future networks. The implementation and results from the evaluation were presented in Paper VI.

Can DPP be used to increase the switch intelligence, enabling processing of complex events in the data plane?

We contributed the design for FastReact, a data plane switch program for complex event detection, along with a feasibility analysis discussing the hardware requirements for such flexibility. We also contributed an actual implementation of the FastReact switch, along with support for alternate path link recovery, using the P4 programming language. This implementation was thoroughly evaluated, using the BMv2 software switch, in an industrial automation network topology. The evaluation demonstrated that FastReact both could reduce the sensor-actuator latency and rapidly cope with link failures. The design, implementation and evaluation results were presented in Paper IV. In Paper VII, we proposed an improved implementation of the FastReact switch, which extended the switch with support for selective multicasting and concurrent packet processing. We increased the flexibility of FastReact, by writing a Python program that could be used to tailor FastReact to the specific device, allowing for improved performance. A detailed evaluation was carried out using the Netronome Agilio SmartNIC platform, which showcased the performance impact of in-network complex event detection. Additionally, we conducted a set of measurements using the T_4P_4S multi-target P4 compiler. Altogether, these measurements demonstrate the feasibility of using FastReact to enable complex event detection in the data plane.

6 Summary of Appended Papers

This section summarizes the appended papers.

Paper I - QoS Enabled WiFi MAC Layer Processing as an Example of a NFV Service

CloudMAC [124] splits the WLAN MAC layer processing and moves the management and data frame processing to a VAP in the cloud. The WLAN APs are simplified into WTPs, and serve as simple radio heads: transmitting the traffic they receive from the VAP to the wireless medium, and forwarding all received wireless traffic to the VAP. Between the WTPs and the VAPs, a SDN-enabled network connects the WTPs to the VAPs. The SDN controllers in this network continuously monitor the signal strength of each station connected to the CloudMAC network and update the forwarding rules between the WTPs and VAPs, when the signal strength weakens. This performs a seamless handover between the APs, as the connection state is kept in the VAP.

In Paper I, we investigated the impact of cross-traffic on WLAN management frames in the CloudMAC network; in particular, we studied the probe response/request frames which are essential for a successful association attempt. This impact is caused by Wireless Network Interface Cards, ignoring WLAN networks that do not respond to probe requests within a certain deadline. Different manufacturers have different delay tolerances for probe responses.

Our investigation showed that in a network where the wired part is heavily congested, the increased probe response delay did increase the prevalence of connection failures. As a solution, we proposed reducing the delay of CloudMAC management frames by prioritizing them in a congested network, using several of the traffic control strategies available in Linux-based switches and routers. We extended Open vSwitch to include support for multiple additional traffic control strategies, such as SFQ [96], CoDel [125], FQ-CoDel [97] [3], and made this available through the SDN controller, using the `ovsdb` [100] interface.

To verify that the added traffic control algorithms would improve the connection success rate, we performed an evaluation using the above-mentioned algorithms and the HTB [95] traffic control algorithm (already implemented in Open vSwitch). The evaluation was performed by loading the SDN-based network with cross-traffic and at the same time trying to connect to a CloudMAC-controlled WLAN station. This showed that the added traffic control algorithms would increase the network reliability from 50% to almost 100% under high load.

Paper II - Resilient Software Defined Networking for Industrial Control Networks

Transmission reliability and fast data delivery are important aspects of Industrial Control Networks (ICNs), where the consecutive loss of data could have severe consequences for a particular industrial process. Providing latency and packet delivery rate guarantees are difficult in Ethernet, especially when repair times have to be below 10 ms.

In Paper II, we proposed using SDN to improve the resiliency of Industrial Sensor Actuator Networks (ISANs). Due to the centralized view of the network, enabled through an SDN-based architecture, link failures can be repaired as soon as the controller receives information on the link status. However, while the link detection in SDN-enabled switches such as Open vSwitch is often slow, the reaction time can be significantly reduced by utilizing the new features in OpenFlow 1.1, such as fast failover and group tables. In fact, under beneficial conditions, the reaction time can be as low as (3.3 ± 0.8) ms [57]. Another method that can reduce the Packet Loss Rate (PLR), is the duplication of packets at the sensor network gateways. Although effectively halving the available link throughput, the packet duplication prevents the loss of packets in the interval between the failures and the failure detection.

An emulation-based testbed implemented in the CORE emulator was used to evaluate the impact of using fast failover in ICNs. The testbed was configured with both a wireless and a wired part, both experiencing packet loss. Additionally, the wired portion of the network experienced link failures throughout the test. Our experiments show how the PLR varies with different BFD message intervals. The PLRs were high due to the configured PLR in

[3] The changes are available in Open vSwitch mainline since 2016, at commit hash: 677d9158fc0aa16f875198d83c7bd8f87238aed5

the emulated wireless portion of the sensor network. Our results show that a low BFD interval often gives a lower PLR. Furthermore, it shows that the PLR is significantly decreased, when the number of gateways and the number of copies in the network increases. This indicates that using SDN with fast failover and packet duplication could increase the reliability of an SDN-based ICN.

Paper III - Low Frequency Assist for mmWave Backhaul - The Case for SDN Resiliency Mechanisms

To cope with the increased demand for mobile network bandwidth, one approach considered for the 5G standard is utilizing the 60 GHz frequency bands, also called the mmWave bands. However, these high-frequency bands suffer from high levels of signal attenuation, shadowing, and blockage. Moreover, these frequency bands typically require near-line-of-sight (LOS) conditions in order to transmit data with an acceptable packet loss rate. To cope with the path loss, mmWave transmitters employ antenna arrays and beamforming techniques to help improve signal strength. Still, despite these improvements, intermittent objects may cause links to be blocked temporarily, especially at high distances between sender and receiver.

Traditionally, 5G mmWave small cell backhaul networks are closed and proprietary, hindering fast innovation and deployment of new networking technologies. In Paper III, in order to open the backhaul network, we proposed using an SDN-based approach for management of SDN-based backhaul networks. One of the most popular SDN protocols, OpenFlow, enables support for fast failover, which can be used to cope with intermittent link failures. Fast failover allows the configuration of backup paths in the networks, which can be used should the primary path be temporarily down. When BFD is used, monitoring packets are continuously sent over each link, reporting the link status. As the monitoring packets are sent, the link quality is continuously reassessed. In that way, the node can choose whether to transmit through the primary path or one of its configured backup paths.

An experiment testbed was set up, using the CORE emulator, with the extensions used in Paper II. The testbed was further improved by implementing a three-state model suggested from measurements in [126]. This model changes the quality of the link by randomly switching between three states: LOS, non-line-of-sight (NLOS), and outage. The results from the paper suggest that a low BFD interval can improve the packet loss of a 5G mmWave small cell backhaul network.

Paper IV - FastReact: In-Network Control and Caching for Industrial Control Networks using Programmable Data Planes

With the recent introduction of IoT and Industry 4.0, the area of industrial automation has gained a lot of attention. This increase in attention has put pressure on current ICNs to expand and provide cloud-enabled highly flexible networking. However, at the core of ICN, the stringent requirements for both

near-absolute reliability and low latency remain. In Paper II, we introduced the idea of using SDN as a provider of resiliency for industrial automation networks, and in this paper we continue exploring what optimizations are possible using the most recent developments in SDN.

This paper presents *FastReact*, which allows outsourcing part of the industrial automation logic (typically performed by a Programmable Logic Controller [PLC]) to the network fabric. This is possible through the use of programmable data planes, which allows re-programming the fast path of a network switch. *FastReact* parses incoming event messages from sensors, records the event data, and can trigger highly customizable actions. These actions are installed from the control plane through the use of Conjunctive Normal Form (CNF)-based tables. Additionally, *FastReact* supports a resiliency technology similar to that provided by BFD and fast failover. *FastReact* could thus be used in future SDN-enabled industrial automation networks, where it can provide even greater control to the SDN orchestrator. Here the orchestrator can decide, for each specific automation process, which events benefits from having in-network logic, and which are better suited for end-host industrial controllers.

To verify the functionality of *FastReact*, we set up a simple experiment testbed, again using the CORE emulator. The emulator had to be extended to support the P4 software switch called BMv2. Results from the experiments show that the switch could both reliably perform fast failover, and was able to reduce the latency of the industrial process, by partially moving the automation logic to the network fabric.

Paper V - IntOpt: In-band Network Telemetry Optimization for NFV Service Chain Monitoring

The concept of INT has recently gained a lot of attention from both the research community and the industry. INT allows network operators to get a highly detailed and up-to-date view of the network state, such as the queue buildup for a certain network path, or the latency for a specific hop. In the INT framework, switches or other network forwarding devices append telemetry information at line rate to data plane packets, as they are forwarded, through the use of a custom INT header. This data is eventually consumed by an INT sink, which sends the traffic to an INT collector, that stores the aggregate data in a network management database, for further processing such as storage, replay or analytics. This data can then be used by the SDN controller to improve forwarding decisions or for offline analysis using e.g., statistical or machine learning methods.

Accurate and timely measurements of network parameters are an important part of operating computer networks. Such measurements can help operators perform quality control, identify and reduce congestion, ensure SLA conformance, and other critical maintenance functions. With the recent developments in NFV, more and more network services are being virtualized, e.g., the Domain Name System (DNS) and Deep Packet Inspection (DPI).

Service Function Chain (SFC) are virtual chains of such Virtual Network Functions (VNFs), which typically are managed by an SDN controller. The SDN controller can then dynamically configure which flows are processed through which SFC. These SFCs may have stringent SLAs imposed on them, so continuous monitoring of network parameters is essential.

Through this work, we proposed IntOpt, a flexible and scalable telemetry collection framework, which monitors a set of SFCs deployed over a substrate network. Through a Simulated Annealing-based Random Greedy (SARG) [127] meta-heuristic approach, the controller determines a set of optimal monitoring flows, such that the total monitoring network overhead is minimized. IntOpt uses this data to determine the placement of INT source, transit and sink nodes, ensuring the required monitoring data is collected with a minimal performance impact. Through benchmarks on the P4-NetFPGA platform and the Open Source Network Tester (OSNT) traffic generator, we benchmark IntOpt and compare it to a naïve approach.

Paper VI - Programmable Event Detection for In-Band Network Telemetry

Traditionally, INT collection, i.e., the INT packet processing, is done by stream processors (e.g., Apache Flink) running on specialized machines. This approach is typically not scalable to line rate processing. One approach which improves scalability of INT collection is INTCollector [30]. It performs INT event detection in the data plane at the INT collector using XDP. Our approach further improves this by performing the event detection directly in the INT sink. Packets are filtered in the network fabric, reducing the INT traffic which reaches the end-host stream processor. We develop a solution running at near line rate and providing an interface to configure flexible event detection, in the P4 data plane, with a per-flow granularity. We also introduce the idea of using FastReact logic processing to further increase configuration flexibility.

We implemented a P4 switch application which processes incoming INT metadata and performs event detection, using the Netronome Agilio NFP4000 hardware. Also, we developed an `AF_XDP` program, that takes event reports from the P4 INT sink and converts them into Apache Kafka messages, which can be sent to a Kafka cluster. Tests were performed through the OSNT NetFPGA application, using the *osnt-extmem* configuration. The tests suggest that the event detection in the data plane improves the potential throughput by several magnitudes and that the `AF_XDP` collector improves throughput further, as data plane processing is performed in- or close to the hardware, removing the need for copying data between kernel- and userspace.

Paper VII - In-Network Caching and Control for SDN-Based Industrial Automation Publish/Subscribe Networks

As a direct follow-up to Paper IV, we implemented FastReact on real P4 hardware and applied it to OPC Unified Architecture (OPC-UA), a real use-case in

industrial networking. OPC-UA [128], introduced by the OPC Foundation, is an effort to standardize the diverse network standards and protocols used in industrial automation. While traditionally proposed as a client-server protocol, the publish/subscribe model has recently been proposed to reduce the network load. With the publisher/subscriber model, sensors act as publishers, which periodically transmit data to an OPC-UA server. This server, in turn, relays the data to any interested subscribers, which typically are actuators or other control systems.

One way to potentially improve the performance and scalability of the OPC-UA server could be to provide the same functionality as provided by an in-network application, in FastReact. In this paper, we implemented the FastReact design on the Netronome Agilio NFP4000 platform and extended it to support multi-threaded packet processing. The NFP-based platforms allow the implementation of functionality as external primitives, using the Micro C language. We implemented FastReact using both a pure P4 approach, as well one utilizing Micro C for the decision logic. We also implemented the same functionality, using the DPDK-based T_4P_4S [44] software switch. Additionally, we implemented a Python code generator, which generates P4 code according to configurable parameters such as memory size and maximum complexity bounds.

To evaluate the performance of FastReact on physical hardware, and its characteristics in an OPC-UA deployment, we configured a simple testbed using the OSNT NetFPGA application. To compare the performance of FastReact with a typical user-space stream processor, we implemented similar CEP functionality to FastReact in Apache Flink, using the FlinkCEP library. From the evaluation, we observe that both tested configurations of FastReact perform significantly better than Apache Flink, due to the reduction of overhead from operating system packet processing.

7 Conclusions and Future Work

The introduction of SDN has brought greater flexibility in the configuration and management of computer networks, and an open protocol along with a large set of open source networking software. From the limitations in traditional SDN, DPP was introduced, to further open up the network device by exposing a programmable data plane. With networking trends pushing for even higher data rates, greater availability, and lower latency, while still requiring strong reconfigurability and flexibility, these two technologies emerge as prominent solutions for providing such flexibility. Through this thesis, we have investigated, implemented, and evaluated solutions for a set of SDN and DPP use-cases, with a belief that the same tools and ideas could be applied to any networking deployment.

This thesis investigated how SDN can be used to increase in-network flexibility and intelligence, improving upon resiliency, monitoring, and control capabilities. First, we investigated how the control-plane can be used to accomplish this, looking in particular at issues of network resiliency. We utilized

both resiliency and traffic control provided by SDN to improve connection establishment in Split-MAC networks, especially under high network congestion. We used fast failover combined with BFD, to improve the packet delivery in both SDN-based industrial automation and 5G mmWave small cell backhaul networks.

In addition to investigating the improvements SDN can have on network resiliency, we used the P4 programming language to design, implement, and evaluate FastReact, a switch program which provides highly configurable in-network event processing, while at same time providing similar resiliency as fast failover combined with BFD. This kind of event processing can both reduce latency, by performing event detection early in the network path, and increase scalability as events can be processed at line-rate directly in the switch hardware, reducing the load on end-hosts. We investigated two separate use-cases for this kind of in-network event detection: industrial automation and network monitoring. Through the use of FastReact, we were able to increase the switch intelligence in SDN-based industrial automation networks, while still preserving the high resiliency mandated by such networks. This increase in intelligence was shown to effectively reduce the sensor-actuator travel path, thus reducing latency and further increasing resiliency. The FastReact implementation was also demonstrated on physical hardware, showing good performance compared to other state-of-the-art CEP applications. Finally, we used our new expertise in DPP to investigate the performance costs of programmable INT, and optimal placement of INT source, transit, and sink nodes for active probing of service chains. These results also guided our XDP INT sink implementation, which enabled high-performance INT event filtering and processing.

Through this thesis, several aspects of using SDN and DPP in industrial networks, Split-MAC WLANs, 5G mmWave small cell backhaul networks, and data center networks, have been analyzed, evaluated, and improved. Taken together, the results obtained strengthen the idea of using SDN and its DPP extensions as an enabler of future, highly flexible, converged networks.

While our investigation has demonstrated the viability of SDN as an enabler of converged networks, there are still a number of unexplored avenues for future research. One example of important future work, is further investigating Split-MAC networks. This includes evaluating how Split-MAC networks manage to uphold low latency in a multi-tenant cloud, where a large number of virtual machines may introduce delays due to context switching. Also, programmable data planes, like P4, could allow us to significantly increase the handover speed, as the handover decision could be taken locally, in the switch. We have also discussed the idea of moving our INT collector to a pure P4 implementation, instead of splitting it between P4 and `AF_XDP`. This could further increase the performance and further simplify deployment in a real network. Finally, in our evaluation of FastReact we only considered the industrial automation use case. We feel that there may be other, perhaps more interesting, use-cases where complex event detection in the data plane could prove useful.

References

[1] Open Networking Foundation. *Software-Defined Networking: The New Norm for Networks*. Tech. rep. Apr. 2012.

[2] P. Bosshart et al. "P4: Programming Protocol-independent Packet Processors". In: *SIGCOMM Comput. Commun. Rev.* 44.3 (July 2014), pp. 87–95. ISSN: 0146-4833. DOI: 10.1145/2656877.2656890.

[3] D. Bhamare et al. "Optimal virtual network function placement in multi-cloud service function chaining architecture". In: *Computer Communications* 102 (2017), pp. 1–16.

[4] M. Martínez-Díaz and F. Soriguera. "Autonomous vehicles: theoretical and practical challenges". In: *Transportation Research Procedia* 33 (2018), pp. 275–282.

[5] K. Xu, Y. Qu, and K. Yang. "A tutorial on the internet of things: from a heterogeneous network integration perspective". In: *IEEE Network* 30.2 (2016), pp. 102–108.

[6] A. Gupta et al. "Sonata: Query-driven Streaming Network Telemetry". In: *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. SIGCOMM '18. Budapest, Hungary: ACM, 2018, pp. 357–371. ISBN: 978-1-4503-5567-4. DOI: 10.1145/3230543.3230555.

[7] C. Kim et al. "In-band Network Telemetry (INT)". In: June 2016. URL: https://p4.org/assets/INT-current-spec.pdf.

[8] Y. Rekhter, T. Li, and S. Hares. *A Border Gateway Protocol 4 (BGP-4)*. RFC 4271. RFC Editor, Jan. 2006. URL: http://www.rfc-editor.org/rfc/rfc4271.txt.

[9] J. Moy. *OSPF Version 2*. STD 54. RFC Editor, Apr. 1998. URL: http://www.rfc-editor.org/rfc/rfc2328.txt.

[10] J. Postel and J. Reynolds. *Telnet Protocol Specification*. STD 8. http://www.rfc-editor.org/rfc/rfc854.txt. RFC Editor, May 1983. URL: http://www.rfc-editor.org/rfc/rfc854.txt.

[11] J. D. Case et al. *Simple Network Management Protocol (SNMP)*. STD 15. http://www.rfc-editor.org/rfc/rfc1157.txt. RFC Editor, May 1990. URL: http://www.rfc-editor.org/rfc/rfc1157.txt.

[12] T. Ylonen and C. Lonvick. *The Secure Shell (SSH) Transport Layer Protocol*. RFC 4253. http://www.rfc-editor.org/rfc/rfc4253.txt. RFC Editor, Jan. 2006. URL: http://www.rfc-editor.org/rfc/rfc4253.txt.

[13] R. Enns et al. *Network Configuration Protocol (NETCONF)*. RFC 6241. http://www.rfc-editor.org/rfc/rfc6241.txt. RFC Editor, June 2011. URL: http://www.rfc-editor.org/rfc/rfc6241.txt.

[14] C. Sieber et al. “Towards a programmable management plane for SDN and legacy networks”. In: *2016 IEEE NetSoft Conference and Workshops (NetSoft)*. IEEE, June 2016. DOI: 10.1109/netsoft.2016.7502428.

[15] D. Kreutz et al. “Software-Defined Networking: A Comprehensive Survey”. In: *arXiv preprint arXiv:1406.0440* (2014), pp. 1–62. DOI: 10.1109/JPROC.2014.2371999. eprint: 1406.0440. URL: http://arxiv.org/abs/1406.0440.

[16] J. Medved et al. “Opendaylight: Towards a model-driven sdn controller architecture”. In: *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*. IEEE. 2014, pp. 1–6.

[17] Ryu SDN Framework Community. *Ryu SDN Framework*. 2017. URL: https://osrg.github.io/ryu/.

[18] P. Berde et al. “ONOS: Towards an Open, Distributed SDN OS”. In: *Proceedings of the third workshop on Hot topics in software defined networking - HotSDN ’14*. ACM Press, 2014. DOI: 10.1145/2620728.2620744.

[19] Z. Xia et al. “KeySight: A Scalable Troubleshooting Platform Based on Network Telemetry”. In: *Proceedings of the Symposium on SDN Research*. ACM. 2018, p. 20.

[20] N. McKeown et al. *OpenFlow: Enabling Innovation in Campus Networks*. Tech. rep. 2008.

[21] Open Networking Foundation. *OpenFlow Switch Specification Version 1.5.1 (Protocol version 0x04)*. URL: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf.

[22] B. Pfaff et al. “The Design and Implementation of Open vSwitch”. In: *12th USENIX Symposium of Networked Systems Design and Implementation* (2015).

[23] Open Networking Foundation. *OpenFlow Switch Specication: Version 1.0.0 Implemented (Wire Protocol 0x01)*. Feb. 2013. URL: https://www.opennetworking.org/wp-content/uploads/2013/04/openflow-spec-v1.0.0.pdf.

[24] Intel. *Increasing Platform Determinism with Platform Quality of Service for theData Plane Development Kit*. Tech. rep. 2016. URL: https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/increasing-platform-determinism-pqos-dpdk-paper.pdf.

[25] T. Høiland-Jørgensen et al. “The eXpress Data Path: Fast Programmable Packet Processing in the Operating System Kernel”. In: *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies*. CoNEXT ’18. Heraklion, Greece: ACM, 2018, pp. 54–66. ISBN: 978-1-4503-6080-7. DOI: 10.1145/3281411.3281443.

[26] S. Goldshtein. “The Next Linux Superpower: eBPF Primer”. In: Dublin: USENIX Association, July 2016.

[27] M. Karlsson and B. Töpel. “The path to DPDK speeds for AF XDP”. In: *Linux Plumbers Conference*. 2018.

[28] G. Bianchi et al. “OpenState: Programming Platform-independent Stateful OpenFlow Applications Inside the Switch”. In: *ACM SIGCOMM Computer Communication Review* 44.2 (2014), pp. 44–51. ISSN: 01464833. DOI: 10.1145/2602204.2602211. URL: http://dl.acm.org/citation.cfm?doid=2602204.2602211.

[29] *PICOS®The First Two-in-One Open Network Operating System (NOS) Coupling Full Enterprise Support with “Classic” SDN*. Pica8. Mar. 2018.

[30] W. Tu et al. “Bringing the Power of eBPF to Open vSwitch”. In: *Linux Plumbers Conference*. 2018.

[31] M. Paolino et al. “SnabbSwitch user space virtual switch benchmark and performance optimization for NFV”. In: *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*. Nov. 2015, pp. 86–92. DOI: 10.1109/NFV-SDN.2015.7387411.

[32] P. Bosshart et al. “Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN”. In: *ACM SIGCOMM Computer Communication Review* 43.4 (2013), pp. 99–110.

[33] The P4 Language Consortium. *$P4_16$ Language Specification version 1.1.0*. Tech. rep. Nov. 2018. URL: https://p4.org/p4-spec/docs/P4-16-v1.1.0-spec.html.

[34] The P4 Language Consortium. *$P4_14$ Language Specification version 1.0.5*. Tech. rep. Nov. 2018. URL: https://p4.org/p4-spec/p4-14/v1.0.5/tex/p4.pdf.

[35] A. Bas et al. *Behavioral Model (bmv2)*. 2019. URL: https://github.com/p4lang/behavioral-model.

[36] Netronome Systems, Inc. *NFP-4000 Theory of Operation*. Tech. rep. 2016. URL: https://www.netronome.com/static/app/img/products/silicon-solutions/WP_NFP4000_TOO.pdf.

[37] G. Brebner. *P4 for an FPGA target*. 2015. URL: https://p4workshop2015.sched.com/event/3ZQA/p4-for-an-fpga-target.

[38] S. Fowler et al. *p4c*. 2019. URL: https://github.com/p4lang/p4c.

[39] Netronome Systems Inc. *Agilio CX SmartNICs*. 2019. URL: https://www.netronome.com/products/agilio-cx/.

[40] Barefoot Networks. *Tofino - World’s fastest P4-programmable Ethernet switch ASICs*. 2019. URL: https://www.barefootnetworks.com/products/brief-tofino/.

[41] Barefoot Networks. *The World’s Fastest & Most Programmable Networks*. Tech. rep. 2019. URL: https://barefootnetworks.com/resources/worlds-fastest-most-programmable-networks/.

[42] J. W. Lockwood et al. “NetFPGA–an open platform for gigabit-rate network switching and routing”. In: *2007 IEEE International Conference on Microelectronic Systems Education (MSE’07)*. IEEE. 2007, pp. 160–161.

[43] W. Tu, M. Budiu, and “fruffy”. *p4-xdp*. 2019. URL: https://github.com/vmware/p4c-xdp.

[44] P. Vörös et al. “T4P4S: A Target-independent Compiler for Protocol-independent Packet Processors”. In: June 2018.

[45] R. Sedar et al. “Supporting Emerging Applications With Low-Latency Failover in P4”. In: *Proceedings of the 2018 Workshop on Networking for Emerging Applications and Technologies*. NEAT ’18. Budapest, Hungary: Association for Computing Machinery, 2018, pp. 52–57. ISBN: 9781450359078. DOI: 10.1145/3229574.3229580.

[46] T. Kohler et al. “P4CEP: Towards In-Network Complex Event Processing”. In: *Proceedings of the 2018 Morning Workshop on In-Network Computing - NetCompute ’18*. ACM Press, 2018. DOI: 10.1145/3229591.3229593.

[47] K. S. Trivedi, D. S. Kim, and R. Ghosh. “Resilience in Computer Systems and Networks”. In: *Proceedings of the 2009 International Conference on Computer-Aided Design*. ICCAD ’09. San Jose, California: ACM, 2009, pp. 74–77. ISBN: 978-1-60558-800-1. DOI: 10.1145/1687399.1687415.

[48] M. F. Vasconcelos and R. M. Salles. “Resilience in Computer Network Management”. In: *NETWORKING 2012*. Ed. by R. Bestak et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 109–120.

[49] R. Zurawski. *Industrial Communication Technology Handbook, Second Edition*. Industrial Information Technology. CRC Press, 2014. ISBN: 9781482207330. (*Mainly chapter 16: Switched Ethernet in Automation*).

[50] R. Perlman. “An algorithm for distributed computation of a spanningtree in an extended LAN”. In: *Proceedings of the ninth symposium on Data communications - SIGCOMM ’85*. ACM Press, 1985. DOI: 10.1145/319056.319004.

[51] D. Levi and H. D. *Definitions of Managed Objects for Bridges with Rapid Spanning Tree Protocol*. RFC 4318 (Informational). Dec. 2005. URL: http://www.ietf.org/rfc/rfc4318.txt.

[52] A. Giorgetti et al. “Performance Analysis of Media Redundancy Protocol (MRP)”. In: *IEEE Transactions on Industrial Informatics* 9.1 (Feb. 2013), pp. 218–227. ISSN: 1551-3203. DOI: 10.1109/TII.2012.2186584.

[53] H. Kirrmann, M. Hansson, and P. Muri. “prp: Bumpless recovery for highly available, hard real-time industrial networks”. In: *Emerging Technologies and Factor Automation* (Sept. 2007).

[54] M. Rentschler and H. Heine. "The Parallel Redundancy Protocol for industrial IP networks". In: *Industrial Technology (ICIT), 2013 IEEE International Conference on* (Feb. 2013), pp. 1404–1409. DOI: 10.1109/ICIT.2013.6505877.

[55] International Electrotechnical Commission (Ginebra). *IEC 62439-3: Industrial Communication Networks : High Availability Automation Networks. Parallel Redundancy Protocol (PRP) and High availability Seamless Redundancy (HSR).* 3. IEC, 2010.

[56] K. Ranjitha et al. "A Proactive Flow Admission and Re-Routing Scheme for Load Balancing and Mitigation of Congestion Propagation in SDN Data Plane". In: *International Journal of Computer Networks & Communications (IJCNC)* 10.6 (2018), pp. 117–134.

[57] N. L. M. van Adrichem, B. J. Van Asten, and F. A. Kuipers. "Fast Recovery in Software-Defined Networks". In: *Software Defined Networks (EWSDN), 2014 Third European Workshop on* (Sept. 2014), pp. 61–66. DOI: 10.1109/EWSDN.2014.13.

[58] S. Sharma et al. "OpenFlow: Meeting carrier-grade recovery requirements". In: *Computer Communications* 36.6 (2013). Reliable Network-based Services, pp. 656–665. ISSN: 0140-3664.

[59] Open Networking Foundation. *OpenFlow Switch Specication: Version 1.1.0 Implemented (Wire Protocol 0x02)*. Feb. 2011. URL: http://archive.openflow.org/documents/openflow-spec-v1.1.0.pdf.

[60] D. Katz and D. Ward. *Bidirectional Forwarding Detection (BFD)*. RFC 5880 (Proposed Standard). June 2010. URL: http://www.ietf.org/rfc/rfc5880.txt.

[61] R. Pigan and M. Metter. *Automating with PROFINET: Industrial Communication Based on Industrial Ethernet*. Wiley-VCH, 2008.

[62] A. Swales et al. "Open modbus/tcp specification". In: *Schneider Electric* 29 (1999).

[63] B. Galloway and G. P. Hancke. "Introduction to Industrial Control Networks". In: *IEEE Communications Surveys Tutorials* 15.2 (Feb. 2013), pp. 860–880. DOI: 10.1109/SURV.2012.071812.00124.

[64] J. Akerberg, M. Gidlund, and M. Bjorkman. "Future research challenges in wireless sensor and actuator networks targeting industrial automation". In: *Industrial Informatics (INDIN), 2011 9th IEEE International Conference on* (July 2011), pp. 410–415. DOI: 10.1109/INDIN.2011.6034912.

[65] A. Scheer. "Industry 4.0: from vision to implementation". In: *Whitepaper,[Online]* 9 (2015).

[66] L. Tan and N. Wang. "Future internet: The internet of things". In: *2010 3rd international conference on advanced computer theory and engineering (ICACTE)*. Vol. 5. IEEE. 2010, pp. V5–376.

[67] Cisco. *Time-Sensitive Networking: A Technical Introduction*. Tech. rep. 2017. URL: https://www.cisco.com/c/dam/en/us/solutions/collateral/industry-solutions/white-paper-c11-738950.pdf.

[68] P. Varis and T. Leyrer. *Time-Sensitive Networking for Industrial Automation*. Tech. rep. Texas Instruments, Jan. 2018.

[69] *P802.1CB – Frame Replication and Elimination for Reliability*. 2017. URL: https://1.ieee802.org/tsn/802-1cb/.

[70] D. Henneke, L. Wisniewski, and J. Jasperneite. "Analysis of realizing a future industrial network by means of Software-Defined Networking (SDN)". In: *2016 IEEE World Conference on Factory Communication Systems (WFCS)*. May 2016, pp. 1–4. DOI: 10.1109/WFCS.2016.7496525.

[71] H. Sandor et al. "Software Defined Response and Network Reconfiguration for Industrial Control Systems". In: *Critical Infrastructure Protection XI*. Ed. by M. Rice and S. Shenoi. Cham: Springer International Publishing, 2017, pp. 157–173. ISBN: 978-3-319-70395-4.

[72] M. Jacobsson and C. Orfanidis. *Using Software-defined Networking Principles for Wireless Sensor Networks*. 2015. URL: %7Bhttp://www.diva-portal.org/smash/record.jsf?pid=diva2%3A817379&dswid=4016%7D.

[73] E. Schweissguth et al. "Application-aware industrial ethernet based on an SDN-supported TDMA approach". In: *2016 IEEE World Conference on Factory Communication Systems (WFCS)*. May 2016, pp. 1–8.

[74] S. B. H. Said, Q. H. Truong, and M. Boc. "SDN-based Configuration Solution for IEEE 802.1 Time Sensitive Networking (TSN)". In: *SIGBED Rev.* 16.1 (Feb. 2019), pp. 27–32. ISSN: 1551-3688.

[75] S. Schriegel, T. Kobzan, and J. Jasperneite. "Investigation on a distributed SDN control plane architecture for heterogeneous time sensitive networks". In: *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*. June 2018, pp. 1–10. DOI: 10.1109/WFCS.2018.8402356.

[76] M. Böhm et al. "Architectural Design of a TSN to SDN Gateway in the Context of In-dustry 4.0". In: (2018).

[77] M. Agiwal, A. Roy, and N. Saxena. "Next Generation 5G Wireless Networks: A Comprehensive Survey". In: *IEEE Communications Surveys Tutorials* 18.3 (2016), pp. 1617–1655. DOI: 10.1109/COMST.2016.2532458.

[78] J. Wannstrom. *LTE-Advanced*. URL: http://www.3gpp.org/technologies/keywords-acronyms/97-lte-advanced.

[79] W. Roh et al. “Millimeter-wave beamforming as an enabling technology for 5G cellular communications: theoretical feasibility and prototype results”. In: *IEEE Communications Magazine* 52.2 (Feb. 2014), pp. 106–113. ISSN: 0163-6804. DOI: 10.1109/MCOM.2014.6736750.

[80] Huawei. *5G Spectrum Public Policy Position*. URL: http://www-file.huawei.com/-/media/CORPORATE/PDF/public-policy/public_policy_position_5g_spectrum.pdf.

[81] Ericsson. *SoftBank and Ericsson to demonstrate 5G 28GHz*. URL: https://www.ericsson.com/en/press-releases/2017/3/softbank-and-ericsson-to-demonstrate-5g-28ghz.

[82] NTT docomo. *Sēkai shuyō benda to no 5G jikken no gaiyō*. URL: https://www.nttdocomo.co.jp/corporate/technology/rd/tech/5g/5g_trial/.

[83] H. Xu, V. Kukshya, and T. S. Rappaport. “Spatial and temporal characteristics of 60-GHz indoor channels”. In: *IEEE Journal on Selected Areas in Communications* 20.3 (Apr. 2002), pp. 620–630. ISSN: 0733-8716. DOI: 10.1109/49.995521.

[84] R. Santos et al. “mmWave Backhaul Testbed Configurability Using Software-Defined Networking”. In: *Wireless Communications and Mobile Computing* 2019 (2019).

[85] K. Seppänen, J. Kilpi, and T. Suihko. “Integrating WMN Based Mobile Backhaul with SDN Control”. In: *Mob. Netw. Appl.* 20.1 (Feb. 2015), pp. 32–39. ISSN: 1383-469X. DOI: 10.1007/s11036-015-0574-7.

[86] J. Núñez-Martínez, J. Baranda, and J. Mangues-Bafalluy. “A service-based model for the hybrid software defined wireless mesh backhaul of small cells”. In: *Network and Service Management (CNSM), 2015 11th International Conference on* (Nov. 2015), pp. 390–393. DOI: 10.1109/CNSM.2015.7367388.

[87] B. Bellalta et al. “Next generation IEEE 802.11 Wireless Local Area Networks: Current status, future directions and open challenges”. In: *Computer Communications* 75 (2016), pp. 1–25. ISSN: 0140-3664. DOI: 10.1016/j.comcom.2015.10.007.

[88] L. Suresh et al. “Towards Programmable Enterprise WLANS with Odin”. In: *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*. HotSDN ’12 (2012), pp. 115–120. DOI: 10.1145/2342441.2342465.

[89] S. Govindan et al. *Objectives for Control and Provisioning of Wireless Access Points (CAPWAP)*. RFC 4564 (Informational). July 2006. URL: http://www.ietf.org/rfc/rfc4564.txt.

[90] P. Calhoun et al. *Lightweight Access Point Protocol*. RFC 5412. RFC Editor, Feb. 2010.

[91] J. Vestin et al. “CloudMAC - Towards software defined WLANs”. In: *Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM* (2012), pp. 393–395.

[92] P. L. Dordal. *An Introduction to Computer Networks*. Department of Computer Science Loyola University Chicago, 2017.

[93] J. F. Kurose and K. W. Ross. *Computer Networking: A Top-Down Approach*. 6th. Pearson, 2012. ISBN: 978-0-13-285620-1.

[94] 2012, IEEE 802.11ae -. *Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications; A*.

[95] J. L. Valenzuela et al. “A hierarchical token bucket algorithm to enhance QoS in IEEE 802.11: proposal, implementation and evaluation”. In: *Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th* 4 (2004), 2659–2662 Vol. 4. ISSN: 1090-3038.

[96] P. E. McKenney. “Stochastic fairness queueing”. In: *INFOCOM ’90, Ninth Annual Joint Conference of the IEEE Computer and Communication Societies. The Multiple Facets of Integration. Proceedings, IEEE* (June 1990), 733–740 vol.2. DOI: 10.1109/INFCOM.1990.91316.

[97] T. Hoeiland-Joergensen et al. *FlowQueue-Codel*. Mar. 2014. URL: https://tools.ietf.org/html/draft-hoeiland-joergensen-aqm-fq-codel-00.

[98] A. Ishimori et al. “Control of Multiple Packet Schedulers for Improving QoS on OpenFlow/SDN Networking”. In: *2013 Second European Workshop on Software Defined Networks* (Oct. 2013), pp. 81–86. DOI: 10.1109/EWSDN.2013.20.

[99] M. S. Seddiki et al. “FlowQoS : QoS for the Rest of Us”. In: *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*. HotSDN ’14 (2014), pp. 207–208. DOI: 10.1145/2620728.2620766.

[100] B. Pfaff and B. Davie. *The Open vSwitch Database Management Protocol*. Internet Requests for Comments. RFC. Dec. 2013. URL: http://www.ietf.org/rfc/rfc7047.txt.

[101] Netronome Systems, Inc. “P4: Driving Innovation in Server-Based Networking”. In: (). URL: https://www.netronome.com/m/documents/WP_P4.pdf.

[102] X. Jin et al. “NetCache”. In: *Proceedings of the 26th Symposium on Operating Systems Principles - SOSP ’17*. ACM Press, 2017. DOI: 10.1145/3132747.3132764.

[103] R. Nishtala et al. “Scaling Memcache at Facebook”. In: *USENIX NSDI*. 2013.

[104] J. Ousterhout et al. “The RAMCloud Storage System”. In: *ACM Transactions on Computer Systems*. Aug. 2015.

[105] G. Siracusano and R. Bifulco. “Is it a SmartNIC or a Key-Value Store?” In: *Proceedings of the SIGCOMM Posters and Demos on - SIGCOMM Posters and Demos ’17*. ACM Press, 2017. DOI: 10.1145/3123878.3132014.

[106] M. Maaß. “In-Network Complex Event Processing using Advanced Data-plane Programming”. MA thesis. June 2018.

[107] P. Carbone et al. “Apache flink: Stream and batch processing in a single engine”. In: *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering* 36.4 (2015).

[108] T. Jepsen et al. “Packet Subscriptions for Programmable ASICs”. In: *Proceedings of the 17th ACM Workshop on Hot Topics in Networks - HotNets ’18*. ACM Press, 2018. DOI: 10.1145/3286062.3286092.

[109] D. Bhamare et al. “A survey on service function chaining”. In: *Journal of Network and Computer Applications* 75 (2016), pp. 138–155.

[110] D. Bhamare et al. “Efficient Virtual Network Function Placement Strategies for Cloud Radio Access Networks”. In: *Computer Communications* 127 (Sept. 2018), pp. 50–60.

[111] C. Kim et al. “In-band network telemetry via programmable dataplanes”. In: *ACM SIGCOMM*. 2015.

[112] N. Van Tu et al. “INTCollector: A High-performance Collector for In-band Network Telemetry”. In: *2018 14th International Conference on Network and Service Management (CNSM)*. IEEE. 2018, pp. 10–18.

[113] J. Hyun and J. W.-K. Hong. “Knowledge-defined networking using in-band network telemetry”. In: *2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE. 2017, pp. 54–57.

[114] Y. Kim, D. Suh, and S. Pack. “Selective In-band Network Telemetry for Overhead Reduction”. In: *2018 IEEE 7th International Conference on Cloud Networking (CloudNet)*. IEEE. 2018, pp. 1–3.

[115] N. Handigol et al. “I know what your packet did last hop: Using packet histories to troubleshoot networks”. In: *11th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 14)*. 2014, pp. 71–85.

[116] D. Reed. *A Balanced Introduction to Computer Science*. 3rd. Pearson, 2012.

[117] G. Dodig-Crnkovic. “Scientific methods in computer science”. In: *Proceedings of the Conference for the Promotion of Research in IT at New Universities and at University Colleges* (2002), pp. 126–130.

[118] J. Ahrenholz et al. “CORE: A real-time network emulator”. In: *MILCOM 2008 - 2008 IEEE Military Communications Conference* (Nov. 2008), pp. 1–7. ISSN: 2155-7578. DOI: 10.1109/MILCOM.2008.4753614.

[119] Grafana Labs. *Grafana*. 2019. URL: https://grafana.com/grafana.

[120] InfluxData. *InfluxDB*. 2019. URL: https://www.influxdata.com/products/influxdb-overview/.

[121] J. Vestin and A. Kassler. "QoS Management for WiFi MAC Layer Processing in the Cloud: Demo Description". In: *Proceedings of the 11th ACM Symposium on QoS and Security for Wireless and Mobile Networks*. Q2SWinet '15 (2015), pp. 173–174. DOI: 10.1145/2815317.2815343.

[122] J. Vestin and A. Kassler. "Resilient SDN based small cell backhaul networks using mmWave bands". In: *2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)* (June 2016), pp. 1–3. DOI: 10.1109/WoWMoM.2016.7523543.

[123] J. Vestin and A. Kassler. "SDN enabled resiliency in LTE assisted small cell mmWave backhaul networks". In: *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)* (Mar. 2017), pp. 199–202. DOI: 10.1109/ICIN.2017.7899411.

[124] P. Dely et al. "CloudMAC - An OpenFlow based architecture for 802.11 MAC layer processing in the cloud". In: *Globecom Workshops (GC Wkshps), 2012 IEEE* (2012), pp. 186–191.

[125] K. Nichols and V. Jacobson. *Controlling Queue Delay*. Tech. rep. 2012.

[126] M. R. Akdeniz et al. "Millimeter Wave Channel Modeling and Cellular Capacity Evaluation". In: *IEEE Journal on Selected Areas in Communications* 32.6 (June 2014), pp. 1164–1179. ISSN: 0733-8716. DOI: 10.1109/JSAC.2014.2328154.

[127] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. "Optimization by simulated annealing". In: *science* 220.4598 (1983), pp. 671–680.

[128] U. Steinkrauss. *Overview: OPC Unified Architecture*. Tech. rep. Sept. 2010. URL: http://www.ascolab.com/images/stories/ascolab/doc/ua_whitepaper_technicaloverview_e.pdf.

SDN-Enabled Resiliency, Monitoring and Control in Computer Networks

Next generation computer networks aim to provide a single network architecture, which can support any type of service, ranging from high-bandwidth video streaming to low-latency industrial automation. Those services have a wide range of network requirements that must be supported by a single converged network, which puts high requirements on flexibility, interoperability, and resilience.

Recently, Software Defined Networking (SDN) and Network Function Virtualization (NFV) have been proposed as solutions for increased network flexibility. By separating and logically centralizing the network control plane, SDN allows for dynamic control of the network infrastructure. NFV, on the other hand, enables flexibility and scalability through the virtualization and orchestration of network functions.

In this thesis, we investigate how SDN and NFV can be used to make next generation networks more reliable, flexible and programmable. We focus mainly on three different areas: resiliency, monitoring, and control, and how they can be improved upon through using SDN.

ISBN 978-91-7867-074-1 (print)

ISBN 978-91-7867-075-8 (pdf)

ISSN 1403-8099

DOCTORAL THESIS | Karlstad University Studies | 2020:2