Time-Sensitive Software-Defined Networking: A Unified Control-Plane for TSN and SDN

Martin Böhm, Jannis Ohms, Manish Kumar, Olaf Gebauer and Diederich Wermser
Ostfalia University of Applied Sciences, Salzdahlumer Str. 46/48, Wolfenbüttel, D-38302, Research Group Communication Systems, {ma.boehm, jannis.ohms2, m.kumar, ola.gebauer, d.wermser}@ostfalia.de

Abstract

5G is separated in three main use cases: Enhanced Mobile Broadband (eMBB), Massive Machine Type Communications (mMTC) and Ultra Reliable Low Latency Communications (uRLLC). All of them have different requirements for the core network of 5G. The 5G-PPP defines Software-Defined Networking (SDN) as a technology for the 5G core network. The IEEE 802.1 Time-Sensitive Networking (TSN) Task Group has published a draft for TSN in fronthaul networks. TSN enables the deterministic forwarding of traffic. The TSN-based fronthaul and the SDN-based core-network are so far configured separately. This paper presents a concept of a unified control-plane for both technologies, called Time-Sensitive Software-Defined Networking (TSSDN). A network can now consists of TSN and SDN devices. Depending on the connection requirements, either deterministic or non-deterministic paths can be set up. A prototype and a testbed have been implemented to assess TSSDN. Results show that the unified control-plane can establish paths using TSN- and SDN-devices.

1 Introduction

In the context of Industry 4.0, 5G plays a central role. It focuses on *Enhanced Mobile Broadband* (eMBB) for high bandwidth user experience, *Massive Machine Type Communications* (mMTC) for low power devices and *Ultra Reliable Low Latency Communications* (uRLLC) to achieve latencies below 1ms for critical machine communication. *Software-Defined Networking* (SDN) has already been set as a network technology for the core network of 5G [1]. SDN decouples the control-plane from the forwarding-plane to have a centralized unit for traffic management. This programmable network can be easily adapted. SDN only supports non-deterministic communication. The *Quality-of-Service* (QoS) capabilities of SDN can be used for low latency applications like *Voice over IP* (VoIP). The IEEE 802.1 *Time-Sensitive Networking* (TSN) Task Group [2] aims on real-time communication in IEEE 802 networks while the *Deterministic Networking* (DetNet) Working Group [3] focuses on the Layer 3 aspects of real-time communication. The TSN Task Group has published a draft for TSN in fronthaul networks [4]. Currently, there are no flexible technologies with automatic device configuration available for *Real-Time Communication* (RTC) in large-scale networks. Ideally, the network should be able to provide time-sensitive and best-effort communication over an IP-based IEEE 802 network. TSN provides these functionalities by scheduling the transmission of packets based on their traffic class. By the means of precise clock synchronization (e.g. IEEE 1588 [5] *Precision Time Protocol* (PTP)), time-critical traffic is transmitted on the basis of a global transmission schedule.

So far, both network technologies are software-defined and have their own control-plane to dynamically configure the network. The combination of the two control-planes into a single control-plane enables a tighter integration. This paper presents a unified control-plane called *Time-Sensitive Software-Defined Networking* (TSSDN), which offers the ability to configure a mixed network of TSN- and SDN-devices. Depending on the devices on a path, end-to-end communications can either be deterministic (only TSN-devices) or non-deterministic (SDN-devices or a mix of TSN- and SDN-devices). Figure 1 shows a scenario, where a *Cyber Physical System* (CPS) communicates over 5G uRLLC and the 5G core network with a real-time application. The TSN-based fronhaul is connected to the 5G core network which is extended by TSN-devices to enable deterministic and non-deterministic communication. The TSSDN-controller configures all devices to set up the required end-to-end communication.

This paper is structured as follows. First, the related work is discussed in Section II. Section III presents the concept of TSSDN and compares the SDN protocol OpenFlow with the IEEE 802.1 TSN standards. Section IV presents an experimental implementation with a testbed along with a discussion of the results. Finally, Section V concludes and presents future work.

2 Related Work

One approach to connect SDN and TSN is the usage of a gateway [6]. Both individual networks are kept separated. The gateway translates the control-plane traffic and makes sure to not disturb the TSN-site, by only sending traffic in its respective time-slots. TSSDN is a centralized control-plane with no need for an additional hardware gateway to communicate between SDN and TSN.

The term TSSDN was first mentioned by Nayak et al. [7]. The idea was a dynamic network reconfiguration for Cyber-Physical Systems (CPS) in a manufacturing system in a software-defined environment. The separation of

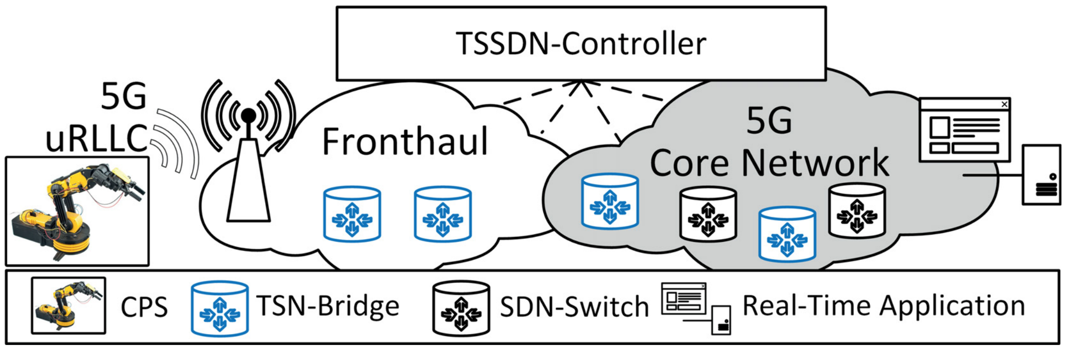

Figure 1 TSSDN in the context 5G

the forwarding (data-plane) and the network configuration (control-plane) of SDN allows the creation of end-to-end communications with user-specified requirements. Real-time communication is not provided in SDN. Hence, the goal of Nayak et al. was a combination of SDN and TSN to support real-time communication, called TSSDN. Their later focus was on the optimization of the real-time scheduler calculation in order to schedule TSN traffic using Integer Linear Programming (ILP) [8, 9]. In contrast, this paper will focus on the integration of SDN (resp. OpenFlow) and TSN over a combined control-plane.

Dürkop et al. presented a high-level architecture for an automatic configuration of real-time Ethernet (RTE) solutions [10]. While Du and Herlich et al. propose the usage of SDN for the network management in a RTE [11, 12], their proof-of-concept implementation shows that some features of the OpenFlow protocol can be used in combination with RTE without any changes to the RTE protocol. For a tighter integration of SDN and other RTE protocols, bigger changes are necessary.

Mizrahi et al. presented *ReversePTP* [13], where each SDN-switch is a master clock, which provides time to the directly connected hosts (PTP clients). The SDN-controller runs multiple instances of a PTP slave and keeps track of the offset of each switch. TSSDN focuses on the control-plane and does not aim to add time-synchronization to SDN-switches. Nevertheless, *ReversePTP* improves the scalability of time-synchronization when compared to the PTP protocol.

This paper presents the concept of an umbrella management (called TSSDN) for the control-plane of TSN and SDN networks. There are no changes on the standards of the two network technologies. Over a centralized control-plane, either deterministic or non-deterministic end-to-end communications can be set up. Furthermore communication over a mixed path of TSN and SDN devices is possible.

3 Architectural Design

This chapter introduces the concept of TSSDN and explains in detail the architecture and resulting problems. It is separated in three planes: the data-plane, the control-plane and the application-plane as specified by the ONF [14]. The concept of TSSDN is explained for each plane and additionally for the exposed interfaces.

3.1 Data-Plane

The data-plane can consists of both, TSN-bridges and SDN-switches.

First of all, OpenFlow SDN-switches need to support the OpenFlow switch specification [15]. It consists of *Flow Tables* which are finite sets of flows. A flow describes how a group of packets are identified, called *match* and how they are processed, called *actionset*. Traffic can *match* on multiple values of certain protocols, from OSI layer 2 to 4. The *actionset* gives actions for matched traffic. This is for example the switch port, on which the packet should be forwarded. It can also be the alteration of header fields. Arriving packets are compared to the *Flow Table* and *actionsets* are performed if the packets *match*. Queuing is provided in OpenFlow for egress bandwidth-reservation and bandwidth-limiting to support QoS. *Meter Tables* are a special type of *Flow Tables* to implement ingress traffic policing by counting and rate limiting packets. *Group Tables* are used for features like load balancing and fast failovers.

In TSN, the bridge needs to support the IEEE 802.1AS-Rev [16] ("Timing and Synchronization for Time-Sensitive Applications") standard for time-synchronization. Thus all TSN-bridges are precisely synchronized. This is important for the time-aware scheduler, IEEE 802.1Qbv [17] ("Enhancements for Scheduled Traffic"), where the processing of different traffic classes is represented in a gate control list (GCL) which opens and closes traffic gates accordingly to implement time-slots. Due to the time-synchronization, the GCL of all TSN-bridges in the network can be optimized. Traffic is filtered with the IEEE 802.1Qci [18] ("Per-Stream Filtering and Policing") standard, which allows filtering and policing of different traffic streams and classes. The bridge is configured by the control-plane. When properly configured, the implementation of the standards allows to have a no wait packet scheduling in the network.

As an interface between the control- and the data-plane, the standard IEEE 802.1Qcp [19] ("YANG Data Model") is used to specify the data model of a TSN-bridge. NETCONF [20] can be used as a network configuration protocol. This interface is used to dynamically update the bridges configuration.

A few TSN standards change aspects of the frame format and media access control of IEEE 802 networks. The IEEE 802.1Qbu [21] standard for "Frame Preemption" al-

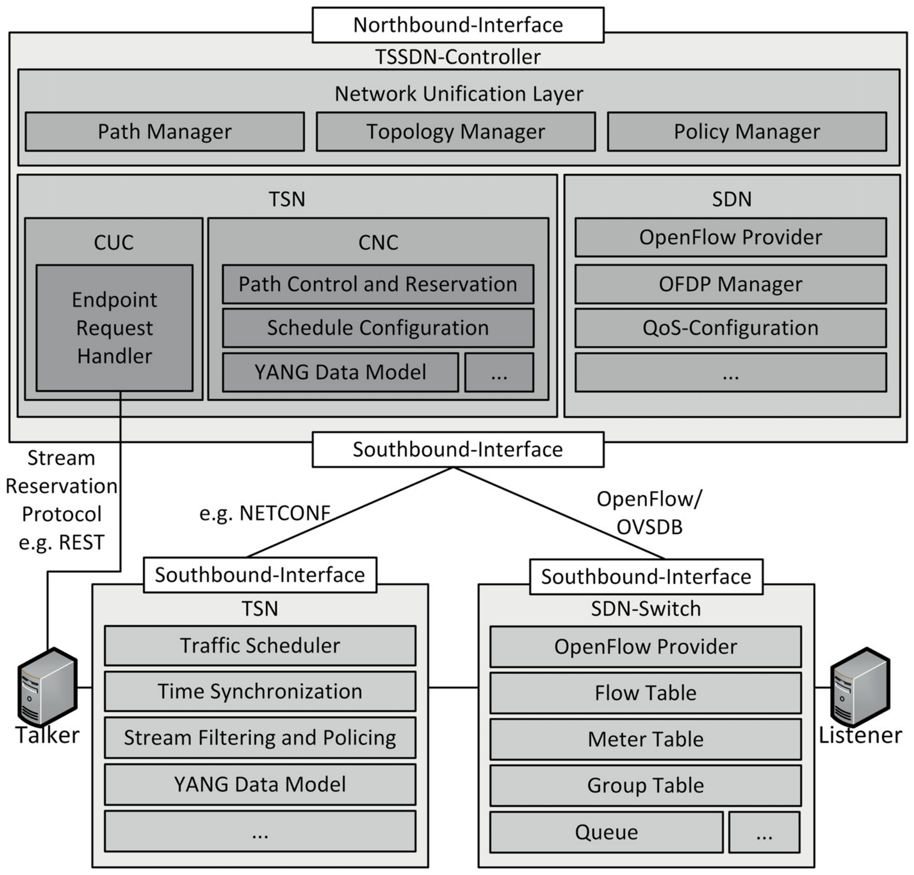

Figure 2 Architecture of Time-Sensitive Software-Defined Networking

lows to interrupt a transmitting frame and resume the transmission later. SDN-switches interpret a preempted frame as corrupted and discard them. Another standard is IEEE 802.1CB [22] for "Frame Replication and Elimination" to replicate frames for redundant transmission on the data-plane including the elimination of the duplicated frames. SDN does not provide a mechanism for the elimination of duplicated frames. The interoperability of TSN and SDN regarding both standards should be investigated separately and is beyond the scope of this paper.

The IEEE 802.1AS-Rev standard for time-synchronization uses a decentralized approach, where a *master clock* is elected by the Best Master Clock Algorithm (BMCA). This approach is against the key property of centralization of SDN. Furthermore SDN is meant to scale larger than a TSN network. The OpenFlow standard does not mention time-synchronization for SDN switches.

3.2 Control-Plane

TSSDN provides a unified umbrella management for deterministic and non-deterministic networks through a combined control-plane.

Due to its architectural location, the control-plane has a view over its network topology including all connected devices. To gather information about the link states, OpenFlow uses the *OpenFlow Discovery Protocol* (OFDP). Here, every switch periodically generates and transmits OFDP-packets on each network port containing its switch id. Whenever another switch receives a OFDP-packet, it is forwarded to the controller. Switches also send copies of ARP-packets to the controller. The controller generates a topology based on these information.

In TSN, the IEEE 802.1Qca [23] ("Path Control and Reservation") standard is used to collect information about the network topology. Furthermore the standard describes routing in a TSN network in case of link failures, path recovery, redundancy, etc.

To realize a connection between two terminal endpoints in SDN, the *FlowTable* of each switch is configured using OpenFlow. Traffic can be identified based on packet headers like IP, TCP, UDP or Ethernet. There are no defined interfaces to establish an end-to-end connection in OpenFlow. Usually, SDN controllers offer a Northbound-API where applications handle the traffic management. These APIs are not standardized. In contrast, a connection in TSN can be requested from a user at the Centralized User Configuration (CUC) (IEEE 802.1Qcc [24]). A user needs to specify his requirements like interval of messages, amount of data, etc. The traffic also needs classification which can be achieved by port numbers, VLAN tags, MAC addresses, etc. to assign traffic to its time-slot. Creating a time-sensitive connection requires information summarized as follows:

1. An overview of the topology
2. A path between source and destination containing only TSN-bridges in case of time-sensitive connections

3. The current scheduler configuration (resp. GCL) of all TSN-bridges on a path
4. The precise time
5. Connection requirements (message interval, amount of data, ...)

With this information a schedule respecting the IEEE 802.1Qch [25] (Cyclic Queuing and Forwarding) standard for each TSN-bridge can be calculated. The device configuration is represented in a YANG data model (IEEE 802.1Qcp [19]).
To avoid inconsistent network states when updating the configurations, Mizrahi et al. presented an approach to timely synchronize the network updates in an SDN [26, 27].
OpenFlow only provides a simple bandwidth based queuing mechanism and *Meter Tables* for rate limiting. Time-based scheduling is not supported. For automated device configuration, companion protocols like *Open vSwitch Database* (OVSDB) can be used. These protocols are not part of the OpenFlow switch specification. OpenFlow also does not provide a mechanism for timely coordinated network updates.
Figure 2 visualizes the architecture of TSSDN. The *Network Unification Layer* orchestrates the interaction between the TSN and the SDN components of the control-plane. It consists of three main modules. The *Path Manager* takes care of the path set up and tear down for each user-specific connection. The *Topology Manager* collects topology information from both network technologies and combines them into one global topology. The last module is the *Policy Manager* which enforces security policies in both networks.

3.3 Interfaces

This section describes interfaces in TSSDN. The control-plane has a southbound interface to communicate with bridges and switches. It is also used to update the devices. The northbound interface on the other side establishes a connection between the control-plane and the application-plane.
OpenFlow standardizes a southbound interface to establish a communication between the OpenFlow switches and the controller. However, there is no standard for the SDN northbound interface. The controller can be extended by applications. A direct user interface between the controller and a terminal endpoint is not provided by any standard.
As already described in section 3.2, a user requests a time-sensitive connection at the CUC using a stream reservation protocol (IEEE 802.1Qcc [24]). For example with the help of a RESTful-API, the user can requests to either create or close a connection, as visualized in Figure 3. It shows that the user sends a request with his requirements to the CUC which is a part of the TSSDN-controller. Using all the information available in the control-plane, the CNC, which is also a part of the TSSDN-controller, calculates a schedule for each TSN-bridge and updates the devices accordingly. Finally, the user is notified about the successful flow setup.
The CUC and the CNC are part of the TSSDN-controller as shown in Figure 2.
In case of a pub/sub-based protocol, for example *OPC-UA-PubSub*, the broker takes care of requesting all connections, because the clients are not aware of their communication partners.

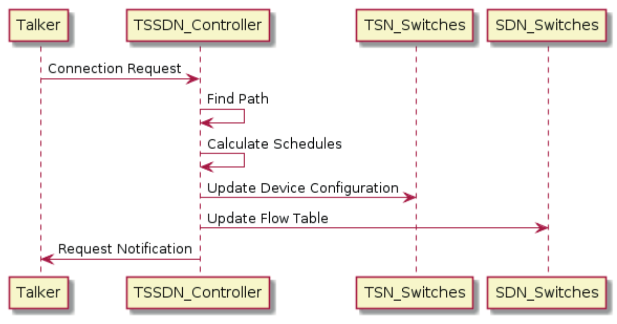

Figure 3 TSSDN connection request sequence diagram

3.4 Application-Plane

The combined control-plane needs a northbound API to provide compatibility with already existing SDN applications. TSN can also benefit from the application-plane, but will further not be discussed in this paper.

4 Implementation

A proof-of-concept prototype of TSSDN has been implemented. A testbed validates and demonstrates the feasibility of the presented approach. We measured the time to request and set up a time-sensitive path. This chapter presents the prototype, describes the evaluation setup and later discusses the results.

4.1 Prototype

The TSSDN implementation is based on the open source SDN-controller Project Floodlight. The concept has been implemented in the form of multiple new modules using the southbound protocols OpenFlow and NETCONF. The set up and tear down of a user-specific (time-sensitive) connection can be requested over the extended Floodlight REST API. The module calculates the shortest path of switches and bridges. A minimal implementation for the schedule calculation has been done. The TSSDN-module generates configurations and updates all related devices.
As a TSN-bridge, the TrustNode platform from InnoRoute [28] has been used which supports time-synchronization, time-aware scheduling and NETCONF for automated device configuration. As an SDN-switch, the Edgecore AS4610-30T switch in combination with the *network operating system* (NOS) PicOS has been used.

4.2 Evaluation Setup and Test Cases

The evaluation topology is shown in Figure 4. It consists of a Talker, a Listener, a TSN-bridge, an SDN-switch, and a TSSDN-controller.

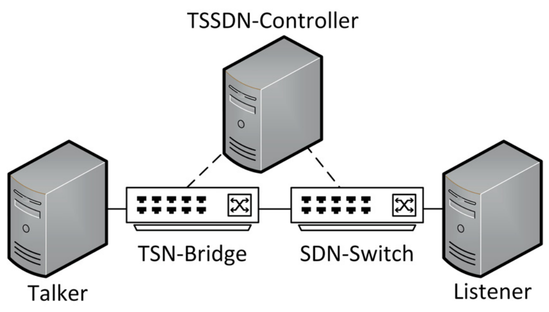

Figure 4 Architecture of the TSSDN testbed

The Talker requests a connection at the REST API of the controller. A configuration for the TSN-bridge is generated based on the connection requirements. The configuration includes a GCL for the time-aware scheduler. A flow is added to the *Flow Table* on the SDN-switch using OpenFlow as already visualized in Figure 3.

For the evaluation, the Talker generates 1000 REST API requests with one request every second. The network traffic of the Talker is recorded. The path set up time is calculated using the HTTP response time, called t_{resp}. In a second test case, the Talker generates traffic using two different UDP-ports. These two traffic classes (TC1 and TC2) are scheduled in the TSN-bridge. The configuration uses a cycle time of 1000ms which is separated in 400ms for TC1 and 600ms for TC2. All the traffic is forwarded to the SDN-switch where a flow is used to forward the traffic to the Listener. Packets are captured on the Listener to validate the expected behavior of the configured scheduler in the TSN-bridge with the actual outcome. It has to be noted, that the combination of TSN and SDN results in a non-deterministic connection.

4.3 Results and Evaluation

For the 1000 REST API requests, to establish a connection between the Talker and the Listener, we measured an average response time of $avg(t_{\text{resp}}) = 314.7ms$, a minimum of $min(t_{\text{resp}}) = 276.3ms$, a maximum of $max(t_{\text{resp}}) = 645.4ms$ and a standard deviation of $\sigma = 17.5ms$.

As the results show, the set up of all devices takes quite some time. This is caused by the schedule calculation and the update of the TSN-bridge and the SDN-switch.

The maximum value always occurs within the first packet due to the just-in-time optimizations of the *Java Runtime* of the controller. An SDN path needs less effort to set up all devices which results in a faster set up time. It has to be noted, that the schedule calculation is simplified in this scenario. The set up time will be affected by the number of hops and the increasing complexity of the schedules.

The results for the second test case are visualized in Figure 5. It shows the desired scheduling behavior has been achieved. Each time-slot starts with a burst of packets which were buffered due to their arrival during the previous time-slot. Just before the end of each full cycle a small gap occurs where no packets are transmitted. This behavior can also be observed when the TSN-bridge is used without an SDN-switch. We are currently investigating this problem with the manufacturer of the TSN-bridge.

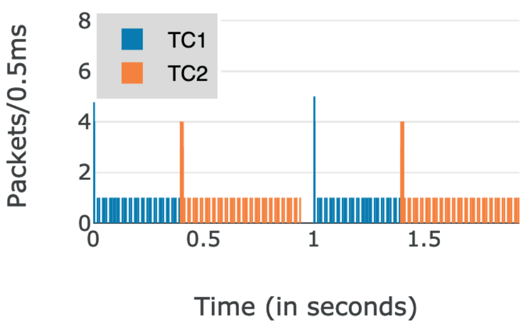

Figure 5 Scheduled packets in a time-sensitive connection

5 Conclusion and Future Work

We presented TSSDN as a concept for an unified control-plane for the network technologies TSN and SDN to create a combined network of TSN-bridges and SDN-switches without any changes to the data-plane. Due to the ability of a flexible configuration, non-time-sensitive connections, as well as time-sensitive connections can be set up on demand. We compared the data-plane, the control-plane, the application-plane and the interfaces for both technologies and proposed an architecture for a unified control-plane. A proof-of-concept prototype has been implemented and a testbed with two clients, a TSN-bridge and an SDN-switch has been set up. Feasibility of the concept has been proven while an average set up time of 314.7ms for a single time-sensitive end-to-end communication is achieved.

As future work, the implementation should be optimized. The scalability of the software needs to be investigated under more realistic conditions using more bridges, switches and time-sensitive connections. A more complex schedule calculation should be used. The applicability of TSSDN for large-scale networks like the 5G core network should be investigated. A connection over multiple TSSDN network operators requires a negotiation protocol to establish connection between the domains of the different TSSDN-controllers.

6 Acknowledgments

This work was partly funded by the Ministry for Science and Culture of Lower Saxony as a part of the research project SecuRIn (VWZN3224) and the Federal Ministry for Education and Research within the KMU-innovativ program as a part of MONAT (16KIS0782).

7 References

[1] 5G PPP Architecture Working Group, “View on 5G Architecture,” *White paper of the 5G-PPP architecture WG (December, 2017)*, 2017.

[2] IEEE 802.1 Working Group. Time-Sensitive Networking (TSN) Task Group. [Online]. Available: https://1.ieee802.org/tsn/

[3] Deterministic Networking (DetNet) Working Group. Deterministic Networking (detnet). [Online]. Available: https://datatracker.ietf.org/wg/detnet/about/

[4] "IEEE Standard for Local and metropolitan area networks – Time-Sensitive Networking for Fronthaul," *IEEE Std 802.1CM-2018*, pp. 1–62, June 2018.

[5] "IEEE Standard Profile for Use of IEEE 1588 Precision Time Protocol in Power System Applications," *IEEE Std C37.238-2017 (Revision of IEEE Std C37.238-2011)*, pp. 1–42, June 2017.

[6] M. Böhm, J. Ohms, O. Gebauer, and D. Wermser, "Architectural Design of a TSN to SDN Gateway in the Context of Industry 4.0," *23. ITG-Fachtagung "Mobile Communications"*, 2018.

[7] N. G. Nayak, F. Dürr, and K. Rothermel, "Software-defined Environment for Reconfigurable Manufacturing Systems," in *2015 5th International Conference on the Internet of Things (IOT)*. IEEE, 2015, pp. 122–129.

[8] N. G. Nayak, F. Dürr, and K. Rothermel, "Time-sensitive Software-defined Networks for Real-time Applications," *IPVS, Uni Stuttgart, Technical Report*, vol. 3, p. 2016, 2016.

[9] N. G. Nayak, F. Dürr, and K. Rothermel, "Incremental Flow Scheduling and Routing in Time-Sensitive Software-Defined Networks," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 5, pp. 2066–2075, 2018.

[10] L. Dürkop, J. Jasperneite, and A. Fay, "An Analysis of Real-Time Ethernets With Regard to Their Automatic Configuration," in *WFCS*, 2015, pp. 1–8.

[11] J. L. Du and M. Herlich, "Software-defined Networking for Real-time Ethernet," in *ICINCO (2)*, 2016, pp. 584–589.

[12] M. Herlich, J. L. Du, F. Schörghofer, and P. Dorfinger, "Proof-of-concept for a Software-defined Real-time Ethernet," in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2016, pp. 1–4.

[13] T. Mizrahi and Y. Moses, "ReversePTP: A clock synchronization scheme for software-defined networks," *International Journal of Network Management*, vol. 26, no. 5, pp. 355–372, 2016.

[14] Open Networking Foundation, "SDN architecture", Tech. Rep. Issue 1, TR-502, 2014. [Online]. Available: https://www.opennetworking.org/wp-content/uploads/2013/02/TR_SDN_ARCH_1.0_06062014.pdf

[15] A. Nygren, B. Pfaff, B. Lantz, *et al.*, "Openflow switch specification version 1.5.1," *Open Networking Foundation, Tech. Rep.*, 2015.

[16] "IEEE Draft Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications," *IEEE P802.1AS-Rev/D7.0, March 2018*, pp. 1–496, Aug 2018.

[17] "IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic," *IEEE Std 802.1Qbv-2015*, pp. 1–57, March 2016.

[18] "IEEE Standard for Local and metropolitan area networks–Bridges and Bridged Networks–Amendment 28: Per-Stream Filtering and Policing," *IEEE Std 802.1Qci-2017*, pp. 1–65, Sep. 2017.

[19] "IEEE Standard for Local and metropolitan area networks–Bridges and Bridged Networks–Amendment 30: YANG Data Model," *IEEE Std 802.1Qcp-2018*, pp. 1–93, Sep. 2018.

[20] A. Bierman and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model," Internet Requests for Comments, RFC Editor, RFC 6536, March 2012.

[21] "IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks – Amendment 26: Frame Preemption," *IEEE Std 802.1Qbu-2016 (Amendment to IEEE Std 802.1Q-2014)*, pp. 1–52, Aug 2016.

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

[23] "IEEE Standard for Local and metropolitan area networks— Bridges and Bridged Networks - Amendment 24: Path Control and Reservation," *IEEE Std 802.1Qca-2015*, pp. 1–120, March 2016.

[24] "IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks – Amendment 31: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements," *IEEE Std 802.1Qcc-2018 (Amendment to IEEE Std 802.1Q-2018 as amended by IEEE Std 802.1Qcp-2018)*, pp. 1–208, Oct 2018.

[25] "IEEE Standard for Local and metropolitan area networks–Bridges and Bridged Networks–Amendment 29: Cyclic Queuing and Forwarding," *IEEE 802.1Qch-2017*, pp. 1–30, June 2017.

[26] T. Mizrahi, E. Saat, and Y. Moses, "Timed Consistent Network Updates in Software-Defined Networks," *IEEE/ACM Transactions on Networking (ToN)*, vol. 24, no. 6, pp. 3412–3425, 2016.

[27] T. Mizrahi and Y. Moses, "Software Defined Networks: It's About Time," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 2016, pp. 1–9.

[28] InnoRoute GmbH. InnoRoute – Innovative Network Equipment and Services. [Online]. Available: https://innoroute.com/