Dynamic Real-Time Stream Reservation for IEEE 802.1 Time-Sensitive Networks with OpenFlow

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

Keywords: Time-Sensitive Networking, Software-Defined Networking, Network Management, Industry 4.0.

Abstract: Industrial network communication requirements are changing within Industry 4.0. Current static industrial networks will require flexibility and need on demand stream reservation with real-time capabilities. Time-sensitive Networking (TSN) offers real-time communication for Ethernet while also providing a mechanism to dynamically request streams (IEEE 802.1Qcc). The standard does not provide concrete specifications for the implementation. This paper evaluates the OpenFlow protocol known from Software-Defined Networking (SDN) for network management in TSN-networks. Requirements for a centralized TSN-controller were identified and OpenFlow has been evaluated if it can fulfill these requirements. An architecture for a TSN-controller has been presented. A proof-of-concept has been implemented and evaluated.

1 INTRODUCTION

Future production facilities are changing within Industry 4.0. New needs emerge for self configuration of network devices. This flexibility allows devices to request their own communication streams. Furthermore, networks have to react on changed configuration while monitoring its health e.g. link failures. These features are not new but getting more demanded when including control and field levels. Current Ethernet-based real-time solutions are often incompatible to standard Ethernet [1] and are also proprietary. As an open standards solution, Time-Sensitive Networking (TSN) guarantees "packet transport with bounded latency, low packet delay variation, and low packet loss" [2] in IEEE 802 networks. The IEEE 802.1Qcc standard [3] specifies on demand TSN stream reservation which is visualized in Figure 1. In combination with the machine to machine communication protocol OPC Unified Architecture (OPC UA), a standard for OPC UA over TSN [4] is currently in standardization. Within the OPC UA PubSub [5] architecture, a centralized communication broker handles the registration of TSN-streams for all communication partners.

The IEEE 802.1Qcc standard does not provide concrete specifications regarding implementation which raises questions for the selection of a User/Network Interface (UNI) protocol for user requests, algorithms for schedule calculation and the selection for a protocol for deploying configurations.

This paper proposes the OpenFlow [6] protocol for the configuration deployment. OpenFlow is commonly used for Software-Defined Networks [7] and already offers a high degree of flexibility. Based on a requirements analysis, an architecture is presented which integrates OpenFlow in the context of TSN. A working prototype has been implemented using an existing open source SDN-controller in accordance with IEEE 802.1Qcc.

This paper is structured as follows. First, the basics of TSN and SDN are presented. Section 3 discusses related work. In Section 4 features and requirements for the in Section 5 presented architecture are described. A proof-of-concept implementation, a testbed and an evaluation is presented in Section 6. Finally, Section 7 concludes and presents future work.

2 BASICS

This chapter gives an overview of the basic functions of TSN. A more detailed view of the IEEE 802.1Qcc standard is given. Later, information about SDN and the OpenFlow protocol are provided.

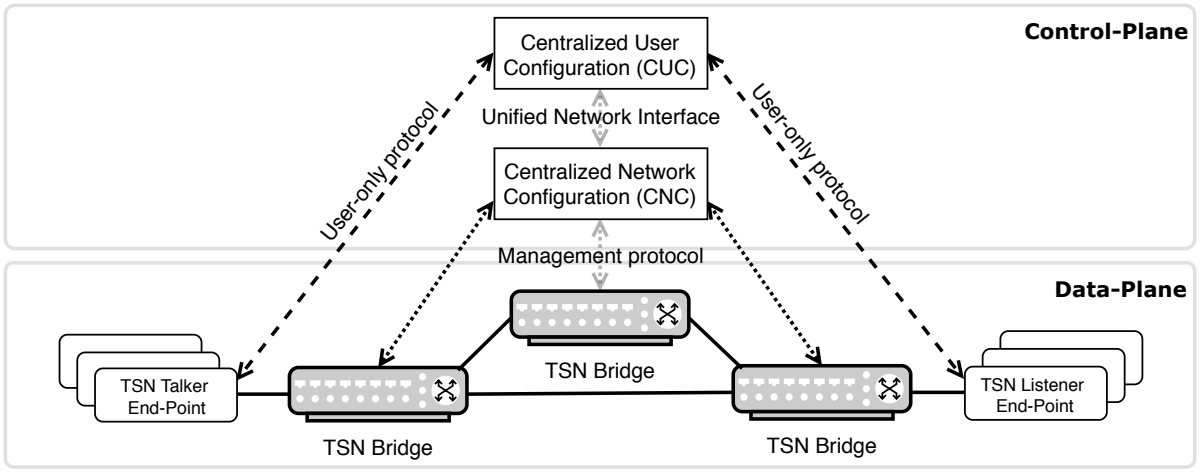

Figure 1: IEEE 802.1Qcc - Fully centralized model.

2.1 TSN Basics

TSN aims on deterministic communication in 802 networks. A bounded latency is achieved by the use of time-slots for network devices (IEEE 802.1Qbv - Enhancements for Scheduled Traffic [8]). Traffic is divided into traffic classes (TC) and assigned to time-slots with cyclical repetition. A configuration is specified in a Gate Control List (GCL). It defines the opening and closing of gates of queues based on the current time. An end-to-end connection in TSN is called a stream. It can for example be identified by the MAC address, IP address or the transport proto-col port. All devices on a TSN-stream path have to be configured properly to transfer frames of a TC. This requires a network-wide precise time-synchronization (IEEE 802.1AS-Rev - Timing and Synchronization for Time-Sensitive Applications [9]). This standard specifies the use of the Precision Time Protocol (PTP) in the context of TSN. Furthermore TSN provides a standard for reliability (IEEE 802.1CB - Frame Replication and Elimination for Reliability [10]), where frames are replicated to be transferred over multiple paths while the duplicated packet is eliminated later. Another standard provides frame preemption (IEEE 802.1Qbu - Frame Preemption [11]), where time-critical frames can suspend the transmission of a non-time-critical frame which will be resumed later.

2.2 Dynamic Stream Reservation

TSN also introduces on demand stream reservation for deterministic streams. An interface for stream requests, a module for schedule calculation for all network devices and the deployment of the configurations are addressed (IEEE 802.1Qcc - Stream Reservation Protocol (SRP) Enhancements and Performance Improvements).

The standard defines three different architectural models for the realization.

1) Fully distributed model: In a decentralized manner without any centralized configuration entities, applications can request their streams directly over the network by propagating the request along the topology using an UNI protocol. Each bridge on a path configures itself with the requirement information given in the request within their limited knowledge of the network.

2) Centralized network/distributed user model: Due to the computational complexity which raises with the amount of devices and streams, a centralized entity, called Centralized Network Configuration (CNC), is introduced. The CNC has global knowledge over all streams and devices in a network. Similar to the fully distributed model, stream requests are send directly over the UNI. The first bridge directs the request to the CNC which configures the bridges after finishing the computation and the generation of the bridges GCLs.

3) Fully centralized model: For more complex use cases, where the talkers and listeners have to be configured too, a Centralized User Configuration (CUC) is introduced as visualized in Figure 1. It discovers end stations and their capabilities, handles application requirements and configures TSN features in the end stations. The CUC forwards the stream information to the CNC using the UNI.

2.3 Software-Defined Networking

Software-Defined Networking (SDN) decouples the data-plane from the control-plane which are conventionally located in the same devices like in

switches and routers. The control-plane defines how frames/packets are forwarded in a devices specific forwarding table, called flow table. In an SDN-domain, all SDN-switches are connected to a logically centralized SDN-controller. Forwarding rules are specified in the application-plane where applications for different purposes decide how to route the traffic. An application can be a shortest path routing application or a firewall. Communication between the SDN-switches and the SDN-controller takes place over the southbound interface. Here, the OpenFlow [6] protocol is dominant and usually supported by all SDN-switches.

3 RELATED WORK

An approach to combine TSN and SDN was first mentioned by Nayak et al. [12]. Their work, called Time-Sensitive Software-Defined Networking, focused on the calculation of schedules using Integer Linear Programming (ILP). Dürkop et al. presented an approach for the automatic configuration of real-time Ethernet (RTE) solutions [13]. Their approach was based on Dynamic Host Configuration Protocol (DHCP). Du and Herlich et al. also proposed the usage of SDN for the network management in RTE [14, 15]. Their proof-of-concept implementation is based on the Powerlink protocol [16]. This RTE implementation works with off-the-shelf switches. Changes on the data-plane are not required. The Powerlink protocol uses a special token to provide deterministic media access. The Powerlink protocol used by Du and Herlich et al. uses different concepts compared to IEEE 802.1 TSN. This paper proposes OpenFlow as a network management protocol for TSN-networks.

4 REQUIREMENTS

This Chapter describes requirements for a TSN-controller. Based on these, OpenFlow is evaluated. Additional features which OpenFlow can not provide are described.

The following requirements were identified for the TSN-controller which includes the CUC and CNC.

- **Topology detection:** The controller needs to be able to detect the topology of the bridges associated.
- **Host detection:** The controller needs to detect each talker and listener in its TSN-network.
- **Time-synchronization:** TSN-bridges, talkers, listeners and the controller need a common time-base (IEEE 802.1As-rev) for the use of IEEE 802.1Qbv.
- **Time-Aware Shaper:** The GCL of each TSN-bridge needs to be calculated and configured.
- **Traffic Classes:** The controller needs to assign Ethernet frames to a queue of the time-aware shaper.
- **Ingress/Egress Policing and Metering:** The controller requires a mechanism to assure that each TSN-stream adheres to the amount of bandwidth it requested.
- **UNI for Talkers/Listeners:** The controller needs to provide a user/network interface (UNI) which offers the ability to request TSN-streams.

Table 1 verifies, if the requirements for the TSN-controller can be fulfilled with the OpenFlow protocol.

OpenFlow does not provide management functionalities for time-aware shapers and also does not provide time-synchronization. For the time-synchronization an additional protocol like Precision Time Protocol (PTP) [17], which is a master/slave protocol, has to be used. The management of the time-aware shapers can either be implemented as a protocol extension of the OpenFlow protocol in the form of experimenter messages or by the use of existing configuration protocols like NETCONF [18]. The UNI for the stream request can be implemented as an extension of the controller. Besides the control-plane, data-plane devices need to support time-synchronization (e.g. PTP) and time-aware shapers.

5 ARCHITECTURE

This chapter presents an architecture for a TSN-controller based on the requirements presented in Chapter 4. The architecture is shown in Figure 2. On the top, it shows the TSN-controller while at the bottom, the functions for a compatible TSN-bridge are shown. Both will further be described.

5.1 TSN-Controller

First of all, the TSN-controller is separated by the CUC and the CNC. The CUC constists of an *Endpoint Request Handler* to provide an UNI which is compatible to IEEE 802.1Qcc [3]. It can be implemented as a REST API. Requests are forwarded from the CUC to the CNC. The *Path Control and Reservation* module has global knowledge about the network and finds paths through the network while re-

TSN-Controller Functions	OpenFlow
Topology detection	OpenFlow networks use LLDP to detect available links.
Host detection	In OpenFlow networks, the controller detects new hosts with the ARP protocol. Each ARP frame received by an OpenFlow-switch is copied and forwarded to the controller.
Time-synchronization	OpenFlow does not provide mechanisms for time-synchronization.
Time-Aware Shaper	OpenFlow provides credit-based shaping to reserve bandwidth for a specific traffic class. There are currently no time-based shapers in the OpenFlow specification.
Traffic Classes	OpenFlow provides an enqueue action which can be used to assign an Ethernet frame to a queue. These queues can be used to implement traffic classes.
Ingress/Egress Policing and Metering	OpenFlow provides ingress and egress metering.
UNI for Talkers/Listeners	OpenFlow does not provide an UNI.

Table 1: OpenFlow protocol feature evaluation for TSN network management.

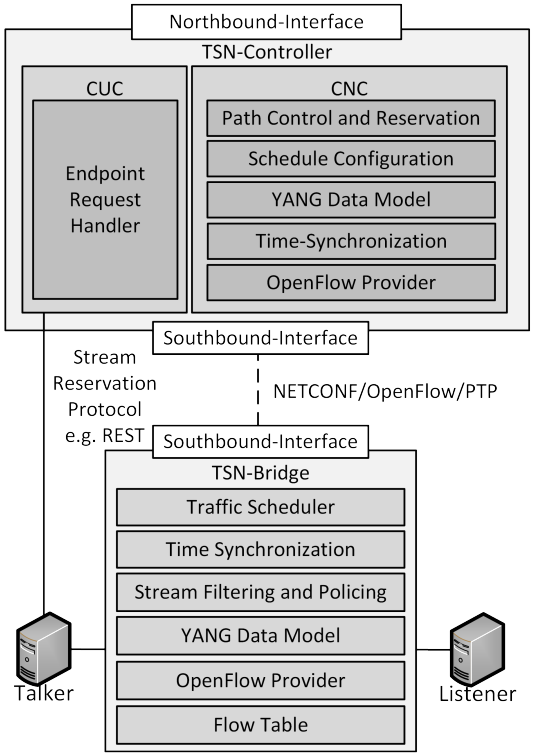

Figure 2: Architecture of the TSN-controller.

specting the TSN-bridges utilization. In the *Sched-ule Configuration* module, a configuration (GCL) for each device on a path is calculated. This process has a very high algorithmic complexity and a lot of research is taking place in this area [19, 20]. The configura-tion has to be represented in a format which can be applied by the TSN-bridges. Here, the *YANG Data Model* is used which is transferred using the *NET-CONF* protocol. For the *Time-Synchronization*, the TSN-controller needs to be part of a PTP-domain. Logically, the master clock should be located in the controller. The *OpenFlow Provider* is used to configure the forwarding behaviour of the TSN-bridges.

5.2 TSN-Bridge

First of all, TSN-bridges need to support a *Traffic Scheduler* resp. IEEE 802.1Qbv. For proper functioning of the *Traffic Scheduler*, the TSN-bridge is timely synchronized. The *Stream Filtering and Policing* module takes care, that TSN-streams do not exceed their requested resources. The TSN-bridge needs to be compatible with a *YANG Data Model* to offer flexible reconfiguration. Over the *OpenFlow Provider*, the TSN-bridge is able to be configured with the OpenFlow protocol. Forwarding behaviour is located in the *Flow Table*.

6 IMPLEMENTATION AND EVALUATION

This chapter describes a proof-of-concept implementation based on the architecture presented in Chapter 5. Later the proof-of-concept will be evaluated.

6.1 Implementation

The proof-of-concept implementation is based on the open-source SDN-controller Ryu [21] written in Python. The data-plane consists of two TSN-bridges (Trustnode) from the company Innoroute which already support IEEE 802.1Qbv, PTP, Netconf and OpenFlow. The Openflow implementation on the bridges is based on Open vSwitch [22]. The talker uses an Intel i210 network card and a kernel extension

to support time stamped packet transmission for TSN-traffic [23]. Except the TSN-briges, all system are based on Ubuntu 18.04 using an Intel Core i7-6700 CPU and 16GB RAM. Every module from Figure 2 is implemented separately. The schedule calculation is simplified due to the complexity of this module. For the CUC interface, the existing REST API from the Ryu controller has been extended.

6.2 Evaluation

To evaluate the proof-of-concept implementation, a test-bed has been set-up as visualized in Figure 3.

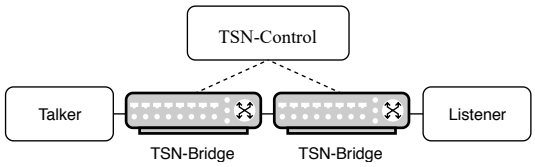

Figure 3: Architecture of the TSN test-bed.

Two aspects of the system have been evaluated. The function and performance of the time-aware shaper and the overall set-up time of a TSN-stream. To test the time-aware shaper, two streams have been requested using two different traffic classes. TC 1 has a time-slot length of 7 ms and TC 2 has a time-slot length of 3 ms resulting in a total cycle length of 10 ms. The talker generates frames at a rate of 100 μs (10 frames per ms) for both TC. Figure 4 shows the arrival of the packets on the listener site. The packets are distributed according to their specified time-slots. Once a time-slot starts, all buffered frames, which arrived outside of their time-slot, are sent.

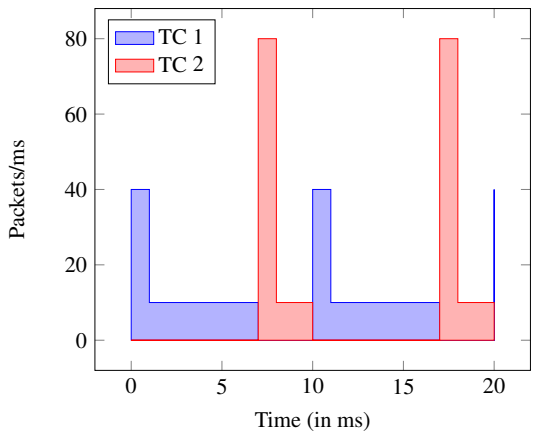

Figure 4: IEEE 802.1Qbv scheduled traffic - 10ms cycle time, 2 traffic classes (7ms and 3ms).

The second measured aspect was the set-up time of the UNI. Here, the talker requests 1 stream every second. Each stream is removed before the next one is created. The UNI protocol request response time, the time between request and response, is measured using Wireshark. For 1000 requests an average set-up time of 3.176 ms with a standard deviation of 1.031 ms has been measured.

The results show, that existing software which is originally developed for SDN can be easily extended to support TSN. It also shows, that TSN-streams can be requested within a few milliseconds. It has to be noted, that the calculation for the schedule is simplified in this proof-of-concept implementation.

7 CONCLUSION

This paper evaluated the use of OpenFlow for an implementation of a TSN-controller with respect to IEEE 802.1Qcc. Requirements for a TSN-controller have been identified, and verified if OpenFlow can fulfill these requirements. Potential extensions and companion protocols have been discussed and an architecture for a TSN-controller has been presented. Later, a proof-of-concept has been implemented with real hardware. The implementation has been evaluated while achieving an average set-up time of 3.176 ms.

OpenFlow itself is only able to partly fulfill the requirements identified. More protocols are needed for a full-featured TSN-controller. The proof-of-concept shows the feasibility of dynamic TSN-stream registrations.

In the future more TSN standards like frame preemption have to be investigated and added to the architecture presented in this paper. OpenFlow should also be considered for the realization of IEEE 802.1CB (Frame Replication and Elimination for Reliability). OpenFlow allows the duplication of frames and the forwarding on multiple output ports.

ACKNOWLEDGEMENTS

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).

REFERENCES

[1] M. Wollschlaeger, T. Sauter and J. Jasperneite, "The future of industrial communication: Automation net-works in the era of the internet of things and industry 4.0," IEEE Industrial Electronics Magazine, vol. 11, no. 1, 2017, pp. 17-27.

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

[3] "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, October 2018.

[4] D. Bruckner, R. Blair, M. Stanica, A. Ademaj, W. Skeffington, D. Kutscher, S. Schriegel, R. Wilmes, K. Wachswender, L. Leurs et al., "Opc ua tsn-a new solution for industrial communication," Whitepaper. Shaper Group, 2018.

[5] OPC Foundation. Part 14: Pubsub. [Online]. Available: https://opcfoundation.org/developer-tools/specifications-unified-architecture/part-14-pubsub

[6] Open Networking Foundation. Openflow switch specification version 1.5.1. [Online]. Avail-able: https://www.opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf

[7] ONF, "SDN architecture," Open Networking Foundation, 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

[8] "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.

[9] "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, August 2018.

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

[11] "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, August 2016.

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

[13] 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.

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

[15] 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.

[16] W. Wallner and J. Baumgartner, "openpowerlink in linux userspace: Implementation and performance evaluation of the real-time ethernet protocol stack in linux userspace," in Proc. 13th Real-Time Linux Workshop (RTLWS).(Prague, Czech Republic, 2011, pp. 155-164.

[17] "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.

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

[19] S.S. Craciunas, R.S. Oliver, and W. Steiner, "Formal scheduling constraints for time-sensitive networks," arXiv preprint arXiv:1712.02246, 2017.

[20] W. Steiner, S. S. Craciunas, and R.S. Oliver, "Traffic planning for time-sensitive communication," IEEE Communications Standards Magazine, vol. 2, no. 2, pp. 42-47, 2018.

[21] Ryu SDN Framework Community. Ryu sdn framework. [Online]. Available: https://osrg.github.io/ryu/

[22] B. Pfaff, J. Pettit, T. Koponen, E. J. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar et al., "The design and implementation of open vswitch." in NSDI, vol. 15, 2015, pp. 117-130.

[23] M. Kumar, M. Boehm, J. Ohms, O. Shulha, and O. Gebauer, "Evaluation of the time-aware priority queueing discipline with regard to time-sensitive networking in particular ieee 802.1 qbv," in Proceedings of International Conference on Applied Innovation in IT, vol. 7, no. 1. Anhalt University of Applied Sciences, 2019, pp. 1-6.