Accepted for presentation in: International Conference on Information and Communication Technology Convergence (ICTC)

Software-Defined Networking driven Time-Sensitive Networking for Mixed-Criticality Control Applications

Fabian Kurtz[1], Gösta Stomberg[2], Maísa Beraldo Bandeira[2], Jens Püttschneider[2], Felix Greiwe, Michael Kaupmann, Christoph Hams, Tim Harnisch, Mey Olivares Tay, Asha Choudhary, Jorge Ramírez Treviño, Abhishek Bhanderi, Apurva Rajashekbhar, Padmashree Vemana, Ahmed Alhanalfi, Timm Faulwasser[2] and Christian Wietfeld[1]

[1]Communication Networks Institute (CNI)

[2]Institute for Energy Systems, Energy Efficiency and Energy Economics (IE3)

TU Dortmund University, 44227 Dortmund, Germany

Email: {fabian.kurtz, goesta.stomberg, maisa.bandeira, jens.puettschneider, christian.wietfeld}@tu-dortmund.de

timm.faulwasser@ieee.org

***Abstract*—Developments such as Industry 4.0, Smart Grids, or Intelligent Transportation System (ITS) depend on reliable high-performance communications to enable the underlying control algorithms. Nevertheless, in most cases it is not viable to provide network infrastructures exclusively for mission-critical control traffic or just the general use case. Hence, 5G as well as future 6G networks entail functionalities such as network slicing to enable the coexistence of mixed-criticality applications on unified networks. An approach capable of supporting slicing in the wireline domain is Time-Sensitive Networking (TSN). It addresses the mentioned challenges by enhancing the Ethernet standard with functionalities required for deterministically bounded low latencies and preemption of high priority data flows. To facilitate its integration with existing 5G and emerging architectures including 6G and open Radio Access Networks (O-RANs), Software-Defined Networking (SDN) has emerged for orchestrating the novel feature set. In this paper, we thus combine TSN and SDN to design an integrated solution. We present experimental results on handling communication-demanding control algorithms and cross traffic simultaneously. Our findings underpin the potential of SDN-driven TSN for mixed-criticality control applications.**

I. Introduction

Modern technology increasingly relies on highly automated systems such as Smart Grids, Intelligent Transportation System (ITS), or Industry 4.0. Their operation necessitates mission-critical control algorithms which in turn depend on reliable communications [1]. Yet, as dedicated networks often are neither available nor viable, these requirements have to be met with shared resources. This leads to mixed-criticality traffic which, e.g., can be realized via 5G [2] network slicing [3, 4]. Here, as well as for future open Radio Access Network (O-RAN) [5] and 6G [6] infrastructures, Time-Sensitive Networking (TSN) has emerged as an technological enabler. First specified via Ethernet [7] amendments, it is standardized in IEEE 802.1Q [8], offering scheduling features via time slots, guard bands, frame preemption, etc. It is particularly relevant in wireline communications, enabling hard service guarantees. Centralized User and Network Configuration (CUC, CNC) serve to manage TSN's features. Yet, integrating TSN into existing networks benefits from harnessing established control planes. Therefore, this work proposes the orchestration of TSN via Software-Defined Networking (SDN), c.f. Fig. 1. To this end, a SDN controller is enhanced for interfacing with TSN-switches. Thereby, the network topology can be monitored and managed centrally. This is evaluated empirically for distributed approaches to automatic control of dynamical systems including communication-demanding Model Predictive Control (MPC). These are deployed on embedded platforms acting as agents. A bottleneck in terms of network capacity is introduced by forcing traffic flows onto a single link and adding best-effort services. This highlights the potential of SDN-driven TSN in challenging mixed-criticality applications such as those found within Smart Grids. This work is structured as follows: After an overview of related works in Sec. II, Sec. III details our approach to SDN-driven TSN. Next, Sec. IV discusses the evaluation scenario, setup and results. Finally, Sec. V draws conclusions and offers an outlook on future work.

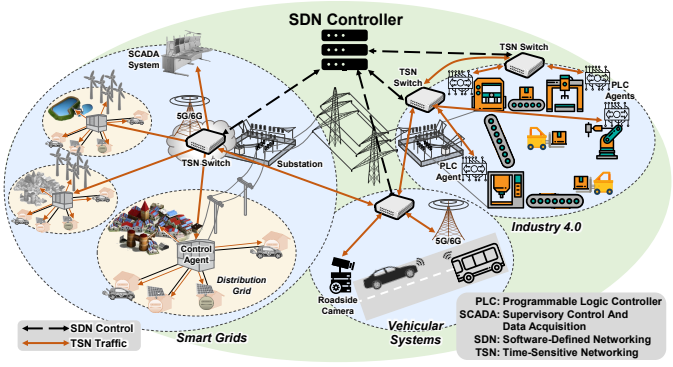

Figure 1: Software-Defined Networking driven Time-Sensitive Networking enables mixed-criticality control applications

II. Related Works

While the orchestration of TSN via SDN has been proposed in related work, several opportunities for further research remain. Existing studies can be grouped into three main categories: Those works primarily concerned with specifying novel architectures for SDN/TSN integration [9, 10], those centered around analytic or simulative means [11–16] and approaches build on empirical testing environments [17, 18]. Regarding works belonging to the first group, the Yet Another Next Generation (YANG) data modeling language as described in an amendment to IEEE 802.1Q for interfacing with TSN devices via network management protocols such as NETCONF or RESTCONF is used in [9, 10]. While providing important contributions to the overall discussion, such works are limited in terms of results supporting the proposed approaches.

Other studies are built around analytic models, e.g., presented by Li et al. [11] for stream reservation and bounded E2E latencies in avionics. Further papers rely exclusively on simulation, often lacking real-world traffic models. For one, the authors of [12] focus on end-to-end latencies for in-vehicular use, while [13] also studies resource usage efficiency as a function of guard band size. In [14] an open source SDN controller, interfacing with TSN via YANG, is coupled to a simulation implemented in the OMNeT++ framework for solving a linear optimization problem. Their focus is placed on assessing various scheduling schemes and their impact on bandwidth utilization. Traffic classification for accurate identification and subsequent prioritization of critical data flows is discussed in [15]. There, OMNeT++ is harnessed to determine end-to-end latencies achievable by the selected approach. Building on this work, the same simulation environment is also applied in a proximate study by the same researchers in [16]. In an effort to ensure a stable performance level in highly dynamic network environments, an SDN-based approach to path reconfiguration is presented.

Finally, research based on measurements performed within testing setups or even real-world environments, such as discussed in this paper, is significantly scarcer. A publication by Thi et al. [17] facilitates the distribution of a highly synchronous timing signal throughout industrial networks. Observed results mostly remain below $500\,\mathrm{ns}$ and as such are suitable for a wide range of applications in the context of Industry 4.0 and beyond. Gerhard et al. [18] employ an open source SDN controller to drive TSN for Open Platform Communications Unified Architecture (OPC UA) applications, but unfortunately do not provide any measurement results owing to critical implementation issues preventing further evaluation of the presented setup.

In contrast, this paper provides an empirical study of SDN/TSN integration on the example of realistic traffic generated by distributed control algorithms. The impact of cross traffic on interframe delays of this mission-critical application is shown to be mitigated, effectively providing the benefits of dedicated networks on shared infrastructures, thereby supporting network slicing and future 5G/6G communications.

III. Software-Defined Network driven Time-Sensitive Networking

Next, we recall the considered distributed algorithms for automatic control which are drawn upon as time-critical communication tasks. Then we detail TSN and our approach.

A. Distributed Control

We design controllers for a set $\mathcal{S} = \{1, \dots, S\}$ of dynamical systems. The dynamics of subsystem/agent $i \in \mathcal{S}$ are

$$\dot{x}_i(t) = \sum_{i \in \mathcal{S}} f_{ij}(x_j(t), u_j(t)), \quad x_i(0) = x_{i,0}, \tag{1}$$

$$y_i(t) = h_i(x_i(t)), \tag{2}$$

where $x_i \in \mathbb{R}^{n_i}$ is the system state, $u_i \in \mathbb{R}^{m_i}$ is the system input, $y_i(t) \in \mathbb{R}^{p_i}$ is the system output, and the functions $f_{ij} : \mathbb{R}^{n_j} \times \mathbb{R}^{m_j} \to \mathbb{R}^{n_i}$ describe dynamic coupling of agents. We apply two different control methods to (1). Firstly, we design networked output-feedback controllers for linear, decoupled, single-input-single-output systems. The control law is

$$u_i(t) = -\sum_{j \in \mathcal{S}} l_{ij} y_j(t),$$

where l_{ij} are the elements of a coupling Laplacian $L \in \mathbb{R}^{S \times S}$, cf. [19]. The matrix L is designed to achieve output synchronization, i.e., $\lim_{t\to\infty} \|y_i(t) - y_j(t)\| = 0 \, \forall i \in \mathcal{S}, \, \forall j \in \mathcal{S}$. This control method requires communication coupled subsystems to exchange messages once per control interval.

As an alternative, we employ Distributed Model Predictive Control (DMPC) (1) to stabilize the systems at $x_i = 0 \; \forall i \in \mathcal{S}$. To this end, we discretize (1) to obtain discrete-time dynamics $f_{ij}^d(\cdot)$. Moreover, we consider compact and convex constraints $\mathbb{U}_i$. In DMPC, the agents at sampling instant $t_k, k \in \mathbb{N}$ measure their current system state $x(t_k)$ and cooperatively solve a discrete-time Optimal Control Problem (OCP) with horizon N of form

$$\min_{x_i, u_i} \frac{1}{2} \sum_{i \in \mathcal{S}} \sum_{k=0}^{N-1} \left(x_i^{k\top} Q_i x_i^k + u_i^{k\top} R_i u_i^k \right) + x_i^{N\top} P_i x_i^N \tag{3a}$$

subject to $\forall i \in \mathcal{S}$:

$$x_i^{k+1} = \sum_{j \in \mathcal{S}} f_{ij}^d \left(x_j^k, u_j^k \right), \quad \forall k \in \{1, \dots, N-1\}, \tag{3b}$$

$$x_i^0 = x_i(t_k), \tag{3c}$$

$$u_i^k \in \mathbb{U}_i, \qquad \forall k \in \{1, \dots, N-1\}. \tag{3d}$$

The agents then apply the input trajectory's first part as control input $u(t_k) = u^0$. OCP (3) is solved by distributed and decentralized optimization methods to obtain a distributed control scheme. Here, we use the Alternating Direction Method of Multipliers (ADMM) and the distributed Active Set Method (ASM) for linear system dynamics and decentralized Sequential Quadratic Programming (dSQP) for nonlinear dynamics, cf. [20, 21]. For a recent overview of other distributed optimization algorithms applicable to DMPC we refer to [20]. Since the applied numerical optimization schemes are iterative in nature, DMPC as a feedback control strategy requires agents to exchange multiple messages per control interval.

B. Foundations of Time-Sensitive Networking

TSN is defined by a set of specifications amended to IEEE 802.1Q [8], in an effort to enable bounded latencies, i.e., deterministic networking, based on the Ethernet standard [7] for time-critical applications. Most relevant for this work are the time-aware shaper, time slots and guard bands originally defined in IEEE 802.1Qbv, as well as preemption (IEEE 802.1Qbu). As shown by the top row of Fig. 2, the time-aware shaper enables scheduling by introducing cyclic slots on the Ethernet network, effectively acting as Time-Division Multiple Access (TDMA). Hence, different traffic classes can be established, allowing the prioritization of critical communication versus other flows such as best-effort data. Nevertheless, issues may arise as this does not allow the interruption of frames currently being transmitted.

Thus, should e.g. a best-effort frame arrive within the tail end of its assigned slot it can intrude on the slot reserved for critical communications, as illustrated by the blue packet above the green slot in Fig. 2 (top row). As such a violation of guarantees is unacceptable, TSN offers guard bands to prevent such situations (c.f. middle row Fig. 2). To ensure that no packet can be send during guard bands, its length has to equal the maximum frame size allowed. However, this approach reduces overall network capacity. Its impact can be minimized via frame preemption as shown in the bottom row of Fig. 2. Here, the ongoing transmission of Ethernet frames can be paused at guard bands, resuming once the assigned periodic time slot reappears. Thereby, wasted link capacity is reduced. Additionally, guard bands can be minimized to the size of a partial packet, increasing usable data rate even further.

In terms of configuration topology TSN employs CNC and CUC. Possible mechanisms such as Management Information Base (MIB) via Simple Network Management Protocol (SNMP) or YANG via NETCONF/RESTCONF are also well established in Ethernet. Optimal integration into existing communication networks can be achieved by building on SDN's powerful and proven set of features for network orchestration.

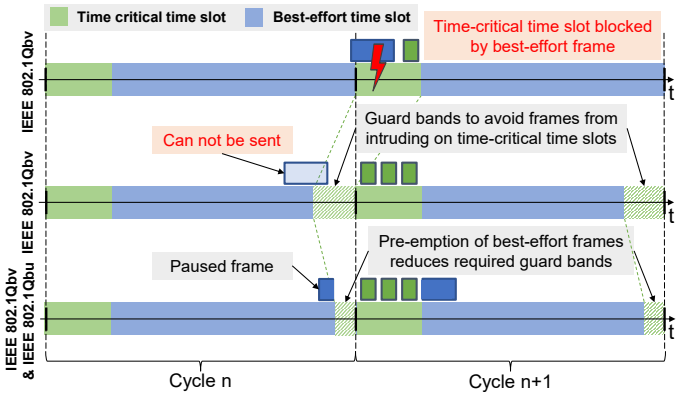

Figure 2: Overview of Ethernet enhancements introduced by TSN as used in this work. Time slots and guard bands specified by IEEE 802.1Qbv and 802.1Qbu preemption enable hard service guarantees for mixed-criticality traffic.

C. Proposed Approach to SDN-driven TSN

Fig. 3 illustrates our concept for orchestrating TSN via SDN. Based on the SDN paradigm, the design can be split into the three layers of application, control, and data plane. Applications, in this case the highly time-critical distributed control algorithms as well as best-effort background traffic, exist on the top level. They sit alongside user-side interfaces such as graphical or command line options, and conceptually can utilize the northbound Application Programming Interface (API) to convey their requirements and commands to the SDN controller, e.g. via Representational State Transfer (REST). This approach hence facilitates application aware network configuration, dynamic communication infrastructure control loops and overall highly automated system operation.

Building on previous works, we here employ a self-developed SDN controller [22] which in turn is forked from the open source Floodlight project [23]. Aside form a broad range of features and intrinsic network monitoring capabilities, we enhanced this entity with support for reconfiguring features offered by TSN. This includes setup and parametrization of functionalities such as guard intervals, time slots of the time-aware shaper and preemption. The SDN controller thus translates commands issued via the northbound API into concrete configurations. By traversing the southbound API, traditionally represented by the de-facto standard OpenFlow protocol, these are transferred to the data plane. At this level, TSN-enabled Ethernet switches handle the physical forwarding of data packets according to the SDN controller's directives. This functionality is enabled by way of a corresponding module, handling SDN/TSN interaction and implemented in this work via REST over Secure Shell (SSH) tunnels. Thereby, bounded deterministic latencies and hard prioritization via time slot scheduling as offered by TSN is orchestrated through our centralized SDN controller. The design also supports 5G network slicing, including on the air interface as demonstrated in [24], thus extending these capabilities to TSN.

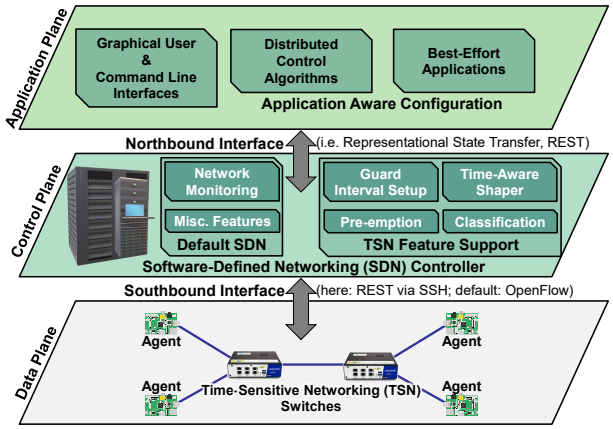

Figure 3: Concept for integrating TSN into SDN-driven communication infrastructures. The SDN controller monitors the network and translates commands issued via a REST-based northbound API into configuration parameters for the TSN-enabled data plane, thereby enforcing hard service guarantees.

IV. Evaluation

This section provides an overview of the evaluation scenario and setup, as well as a detailed discussion of achieved results.

A. Evaluation Scenario & Experimental Setup

The evaluation setup created for the purposes of this work is shown in Fig. 4. Two NXP LS1021ATSN serve as data plane, i.e., TSN switches, offering a per link data rate of $100\,\mathrm{Mbit\,s^{-1}}$. They connect to each other, as well as to embedded computers (i.e., Raspberry Pi 3) acting as distributed control agents. However, unlike its peers Agent 4 does not run control algorithms, but introduces generic best-effort cross traffic into the network, fully saturating the middle link. Thereby, a resource conflict is generated on the central link between both TSN switches. Network orchestration is performed by an SDN controller. It is a self-developed solution used in previous works [25] and based on the open source Floodlight controller [23]. Among its tasks is the configuration of the time-aware shaper to ensure hard service guarantees for control applications in a mixed-criticality environment. The various traffic streams' dissimilar priority levels are encoded into the field offered by Virtual Local Area Networks (VLANs) to specify service classes and acted upon by the switches.

TSN depends on highly synchronized network entities to enable the time slot based concept required for ensuring bounded latencies. Therefore, we employ the Precision Time Protocol (PTP) [26] to establish a firm time reference throughout the setup. Hence, timing variance at the different devices is minimized, allowing stable operation which in turn provides the high confidence levels required for the subsequent evaluation. Observed measurement results are given in Fig. 5. As clearly shown, the TSN switches' hardware assisted PTP implementation yields significantly more homogeneous and precise offsets in contrast to the software solution employed at the agents. While hardware PTP shown on the left mostly exhibits delays in a range of approximately $\pm 100\,\mathrm{ns}$ with peaks well into the $300\,\mathrm{ns}$ range, software PTP fluctuates primarily between $\pm 20\,\mu\mathrm{s}$, with outliers below $100\,\mu\mathrm{s}$. Taking this into account, results presented in the following are rounded to $100\,\mu\mathrm{s}$.

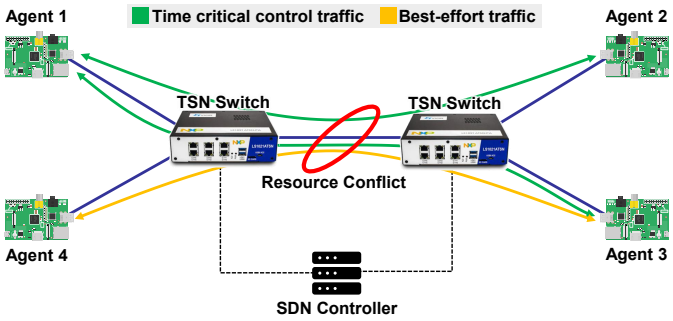

Figure 4: Setup for evaluating SDN-driven TSN comprised of two switches orchestrated by an SDN controller. Four agents execute distributed control algorithms and generate best-effort traffic, thereby creating a resource conflict on the central link.

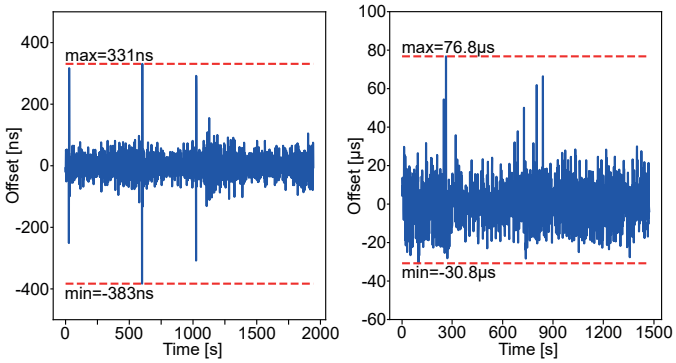

Figure 5: Synchronization of the evaluation setup. Software based (control agents, left) vs. hardware based PTP (TSN switches, right). Note the different orders of magnitude and significantly more stable clocks on the right.

As control applications, one scenario with the output-feedback controller and three scenarios with DMPC are analyzed for networks with three agents ($n_i = 2$, $m_i = p_i = 1$). For the output-feedback controller, we consider a network of fully coupled undamped oscillators as individual subsystems (Scenario 1), design L, and choose a sampling time of $15\,\mathrm{ms}$. In case of DMPC, we design controllers with $N = 5$ and a sampling time of 200 ms for a linear chain of masses system and apply either ADMM (Scenario 2) or ASM (Scenario 3). We use five ADMM iterations per MPC step to solve OCP (3) in Scenario 1 and solve the OCP to optimality with ASM in Scenario 2. Additionally, we design a controller with $N = 4$ and a sampling time of 100 ms for a network of three fully coupled nonlinear Van der Pol oscillators with parameters from [27] and apply dSQP (Scenario 4). In dSQP, we run 25 inner iterations per MPC step to solve OCP (3). We use qpOASES [28] to solve the arising optimization problems in ADMM, CasADi [29] to compute sensitivities, and the Lightweight Communications and Marshalling (LCM) [30] library to exchange iteration variables between agents.

B. Evaluation Results

Fig. 6 shows the measured interframe latencies for the DMPC scenarios. Each of them is analyzed under three different traffic conditions. Black dots in the subgraphs mark the respective mean interframe latencies. The left column displays scenarios with inactive TSN and no cross traffic. Hence, these distributions represent the ideal traffic patterns generated by the employed control algorithms. Any deviation thus implies unwanted interference with application behavior, i.e., as caused by a lack of control traffic prioritization. Therefore, the ideal result achievable by the presented SDN-driven approach to TSN is the avoidance of such variations.

The control applications send messages with an interframe delay of around $1\,\mathrm{ms}$ to solve the OCP. Also, messages with larger interframe delays occur when waiting for the following control interval. This applies in case of ADMM and ASM, whose communication behavior features interframe delays just below $200\,\mathrm{ms}$ respectively slightly above $190\,\mathrm{ms}$, as depicted in the corresponding rows of the leftmost column in Fig. 6.

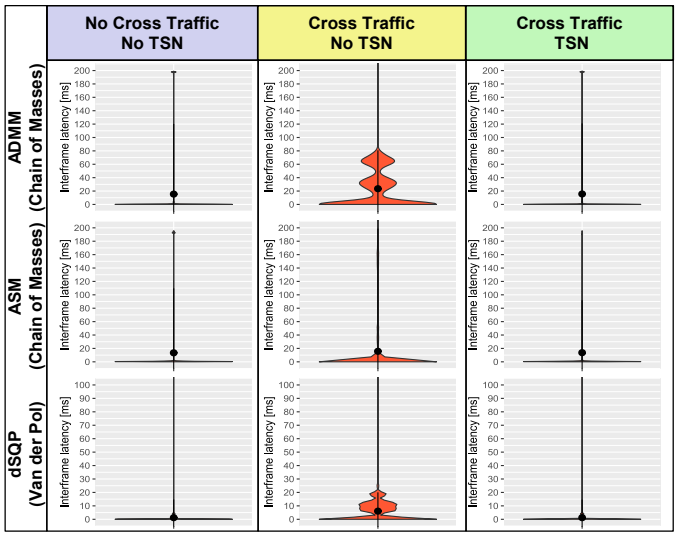

Figure 6: Measured interframe latencies for DMPC. Without TSN (middle row) delay distributions are widened significantly due to the interference of cross traffic. With TSN active (right), performance is identical to a dedicated network (left).

The center column shows results including cross traffic with inactive TSN, i.e., as can be observed in traditional Ethernet based networks. In all three DMPC scenarios, cross traffic increases interframe delays thereby interfering with the various algorithms. As a result, the time required to solve the OCP also increases, which could deteriorate the control performance and thereby impact mission-critical applications. In case of ADMM the distribution widens at slightly above 30 ms and around 65 ms, with individual outliers going beyond 200 ms. ASM is not impacted as strongly. Yet, the overall latencies increase visibly with more values scattered up to 190 ms. Lastly, dSQP forms two bulges from 5 ms to 20 ms. Thus, all three cases illustrate the distinct effect of cross traffic on timely delivery of traffic generated by distributed control agents as caused by resource conflicts (c.f. Fig. 4), typically encountered in shared/sliced (public) communication infrastructures.

Finally, the right column of Fig. 6 presents results with cross traffic and active SDN-driven TSN. In all cases, interframe delay distributions appear virtually identical to those without competing data flows, as shown on the very left. This highlights how TSN ensures the uninterrupted and correct execution of the selected automatic control algorithms despite resource conflicts. Hence, the viability of our setup and its underlying technologies for mixed-criticality applications, as outlined at the start of this work, is demonstrated and validated by way of a realistic application scenario.

Table I: Mean delays observed in the evaluation of DMPC, highlighting TSN's mixed-criticality performance.

	ADMM Chain of Masses	**ASM Chain of Masses**	**dSQP Van der Pol**
	Mean Interframe Delay [ms]	**Mean Interframe Delay [ms]**	**Mean Interframe Delay [ms]**
No Cross Traffic, No TSN	15.4	13.4	1.2
Cross Traffic, No TSN	23.5	15.6	6.1
Cross Traffic, TSN	15.5	13.4	1.2

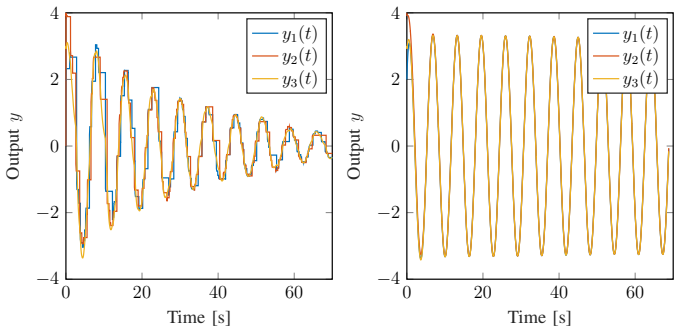

Figure 7: Closed-loop oscillator trajectories with cross traffic and inactive TSN (left) and active TSN (right). The desired synchronous oscillation is only possible with active TSN.

Tab. I summarizes the mean interframe delays for the analyzed DMPC and traffic scenarios. Results are rounded to 100 µs which, as previously discussed, serves to account for clock jitter highlighted in the Fig. 5 PTP chart. Row one and three present identical values, with a very minor variation in ADMM caused by the slightest variance in measurement, which is to be expected in an empirical study. In contract, the middle row shows a clear deviation from optimal results as caused by interfering best-effort communication, thereby aligning with the previously discussed delay distributions. Hence, this further illustrates the effectiveness of SDN-driven TSN to provide hard service guarantees in challenging mixed-criticality communication environments, e.g. shared networks.

In addition to the DMPC scenarios, we also provide an example from output-feedback control to graphically illustrate the impact of TSN - or lack thereof - on control algorithm performance. Fig. 7 shows the closed-loop trajectories of the controlled oscillators for two different network scenarios. On the right hand side of the plot the desired control performance is shown, i.e., output synchronization of the three agents to a common oscillation. This control behavior is facilitated by TSN, which guarantees the timely communication of messages relevant for the control algorithm. Performance is thus effectively identical to what can be measured within a communication network dedicated exclusively to mission-critical application traffic. Conversely, the left of Fig. 7 shows the closed-loop result without TSN. Here, cross traffic interferes with the control application. Thus the desired outcome as represented by synchronized oscillation is not achieved. In summary, this aligns with the performance level expected from and indicated by the previously discussed network delays. This underlines the suitability of our proposed approach to

SDN-driven TSN for supporting and enabling hard service guarantees regardless of interfering traffic. It thus enables wireline network slicing in mixed-criticality use cases within application domains such as Industry 4.0 or Smart Grids.

V. Conclusion and Outlook

In this work, we present an approach to SDN-driven TSN for mixed-criticality control applications such as Industry 4.0 or Smart Grids. For this, TSN is integrated with SDN. Thereby, novel features introduced to Ethernet by IEEE-defined standard extensions forming TSN become available for centralized and application aware configuration via SDN. This includes the time-aware shaper with cyclic slots as well as guard bands and frame preemption. A purpose developed testing environment is presented and serves to empirically evaluate our approach on the example of challenging traffic generated by various distributed real-world control algorithms. Here, a resource conflict between agents performing calculations and exchanging values across a single central link and generic best-effort traffic is created. Measurement results focus on interframe latencies to study the impact of cross traffic on agent communication. Additionally, output trajectories are given to visually illustrate the impact of competing packet flows, or lack thereof, on control. In summary, it is shown that TSN allows mixed-criticality environments to achieve a performance level identical to dedicated networks. Thus, the approach lends itself to realizing wireline network slicing.

Future work will focus on enhancing and extending the setup via integration into a realistic Smart Grid deployment scenario in scope of a cellular energy grid laboratory demonstrator. Moreover, ongoing O-RAN and 6G research will be strengthened by combining our approach with machine learning to continuously improve parametrization. Here, 6G energy efficiency will also be studied in depth.

Acknowledgement

This work has been partly funded by the Federal Ministry for Economic Affairs and Climate Action (BMWK) via the project *5Gain* under funding reference 03EI6018C and is supported by the Federal Ministry of Education and Research (BMBF) via the project *6GEM* (funding reference 16KISK038). The authors thank Henrik Ebel for his contributions to the MPC implementation.

References

[1] D. Kumar and N. Pindoriya, 'A Review on 5G Technological Intervention in Smart Grid,' in *Power Systems Conference*, 2020, pp. 1–6.

[2] 3GPP, 'Technical Specification Group Services and System Aspects; System architecture for the 5G System (5GS); Stage 2; Release 17,' 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 23.501, 2022.

[3] X. Foukas, G. Patounas, A. Elmokashfi and M. K. Marina, 'Network Slicing in 5G: Survey and Challenges,' *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, 2017.

[4] A. Nota, S. Saidi, D. Overbeck, F. Kurtz and C. Wietfeld, 'Providing Response Times Guarantees for Mixed-Criticality Network Slicing in 5G,' in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2022, pp. 552–555.

[5] *O-RAN ALLIANCE Specifications*, 2022. [Online]. Available: https://orandownloadsweb.azurewebsites.net/specifications.

[6] N. H. Mahmood, S. Böcker, I. Moerman, O. A. López, A. Munari, K. Mikhaylov, F. Clazzer, H. Bartz, O.-S. Park, E. Mercier *et al.*, 'Machine Type Communications: Key Drivers and Enablers towards the 6G Era,' *EURASIP Journal on Wireless Communications and Networking*, vol. 2021, no. 1, 2021.

[7] *IEEE Standard for Ethernet*, IEEE Standard 802.3, 2015.

[8] *IEEE Standard for Local and Metropolitan Area Network – Bridges and Bridged Networks*, IEEE Standard 802.1Q, 2018.

[9] S. B. H. Said, Q. H. Truong and M. Boc, 'SDN-Based Configuration Solution for IEEE 802.1 Time Sensitive Networking (TSN),' *SIGBED Rev.*, vol. 16, no. 1, pp. 27–32, 2019.

[10] H. Chahed and A. J. Kassler, 'Software-Defined Time Sensitive Networks Configuration and Management,' in *IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 2021, pp. 124–128.

[11] E. Li, F. He, L. Zhao and X. Zhou, 'A SDN-based Traffic Bandwidth Allocation Method for Time Sensitive Networking in Avionics,' in *IEEE Digital Avionics Systems Conference (DASC)*, 2019, pp. 1–7.

[12] T. Hackel, P. Meyer, F. Korf and T. C. Schmidt, 'Software-Defined Networks Supporting Time-Sensitive In-Vehicular Communication,' in *IEEE Vehicular Technology Conference (VTC2019)*, 2019, pp. 1–5.

[13] H. Lee, J. Lee, C. Park and S. Park, 'Time-aware Preemption to Enhance the Performance of Audio/Video Bridging (AVB) in IEEE 802.1 TSN,' in *IEEE International Conference on Computer Communication and the Internet (ICCCI)*, 2016, pp. 80–84.

[14] M. A. Metaal, R. Guillaume, R. Steinmetz and A. Rizk, 'Integrated Industrial Ethernet Networks: Time-sensitive Networking over SDN Infrastructure for mixed Applications,' in *IFIP Networking Conference (Networking)*, 2020, pp. 803–808.

[15] N. S. Bülbül, D. Ergenç and M. Fischer, 'SDN-based Self-Configuration for Time-Sensitive IoT Networks,' in *IEEE Conference on Local Computer Networks (LCN)*, 2021, pp. 73–80.

[16] ——, 'Towards SDN-based Dynamic Path Reconfiguration for Time Sensitive Networking,' in *NOMS IEEE/IFIP Network Operations and Management Symposium*, 2022, pp. 1–9.

[17] M.-T. Thi, S. Ben Hadj Said and M. Boc, in *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*.

[18] T. Gerhard, T. Kobzan, I. Blöcher and M. Hendel, 'Software-defined Flow Reservation: Configuring IEEE 802.1Q Time-Sensitive Networks by the Use of Software-Defined Networking,' in *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2019, pp. 216–223.

[19] F. Bullo, *Lectures on Network Systems*, 1.6. Kindle Direct Publishing, 2022. [Online]. Available: http://motion.me.ucsb.edu/book-lns.

[20] G. Stomberg, A. Engelmann and T. Faulwasser, 'A compendium of optimization algorithms for distributed linear-quadratic MPC,' *at-Automatisierungstechnik*, vol. 70, no. 4, pp. 317–330, 2022.

[21] ——, 'Decentralized non-convex optimization via bi-level SQP and ADMM,' *arXiv preprint:2204.08786*, 2022.

[22] F. Kurtz, I. Laukhin, C. Bektas and C. Wietfeld, 'Evaluating Software-Defined Networking-Driven Edge Clouds for 5G Critical Communications,' in *International Conference on Information and Communication Technology Convergence (ICTC)*, Jeju, Korea, 2018, pp. 405–410.

[23] *Floodlight SDN OpenFlow Controller Version 1.2*, 2022. [Online]. Available: https://github.com/floodlight/floodlight.

[24] F. Kurtz, R. Wiebusch, D. Overbeck and C. Wietfeld, 'Predictive 5G Uplink Slicing for Blockchain-driven Smart Energy Contracts,' in *IEEE Conference on Communications Workshops*, 2022, pp. 19–24.

[25] N. Dorsch, F. Kurtz and C. Wietfeld, 'Enabling Hard Service Guarantees in Software-Defined Smart Grid Infrastructures,' *Computer Networks*, vol. 147, pp. 112–131, 2018, Elsevier.

[26] *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Std 1588, 2019.

[27] D. Burk, A. Völz and K. Graichen, 'A Modular Framework for Distributed Model Predictive Control of Nonlinear Continuous-Time Systems (GRAMPC-D),' *Optimization and Engineering*, vol. 23, no. 2, pp. 771–795, 2022.

[28] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock and M. Diehl, 'qpOASES: A parametric active-set algorithm for quadratic programming,' *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[29] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings and M. Diehl, 'CasADi: a software framework for nonlinear optimization and optimal control,' *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[30] D. Moore, E. Olson and A. Huang, 'Lightweight Communications and Marshalling for Low-Latency Interprocess Communication,' Massachusetts Institute of Technology - Computer Science and Artificial Intelligence Laboratory, Technical Specification (TS) MIT-CSAIL-TR-2009-041, 2009.