# Integrated Industrial Ethernet Networks: Time-sensitive Networking over SDN Infrastructure for mixed Applications

Mohamed Abdel Metaal[1], Rene Guillaume[2], Ralf Steinmetz[1], and Amr Rizk[1,3]

[1]Technische Universität Darmstadt
[2]Corporate Research and Advanced Development, Robert Bosch GmbH
[3]Universität Ulm

***Abstract*—A unified network architecture, allowing for simultaneous transmission of multiple real-time (RT) and non-RT traffic classes, is becoming increasingly important for the realization of industrial IoT. There is yet no clear approach allowing multiple state-of-the-art Industrial Ethernet applications, exhibiting different isochronous hard real-time traffic patterns, to share the same network resources alongside best-effort traffic.**

**This paper presents an approach to integrating heterogeneous Industrial Ethernet applications over Time-Sensitive Networks (TSN). We deploy different real-time schedulers that are embedded in a Software-defined Network (SDN) controller to allow different applications with various traffic patterns and Quality-of-Service (QoS) requirements to share the same network. This integration is accomplished by providing an abstraction that combines the requirements of the Industrial Ethernet application and the TSN network capabilities and constraints.**

***Index Terms*—Time Sensitive Networking (TSN), Software-defined Networking (SDN), Quality of Service (QoS).**

## I. Introduction

Cyber-physical systems comprise highly time-sensitive applications, typically demanding zero packet loss and strict real-time guarantees with bounded latency and jitter. These systems depend on communication networks to deliver control commands and sensor data, from the controllers to sensors and actuators and back. Such systems are considered strictly time-sensitive as network delays and jitter strongly impact their functionality. A typical example are Motion Control Systems (MCS) that require isochronous communication between its controller (master) and its servo/motor drives (slaves).

While traditional field-bus networks comprise different limitations including no support of high data rates, a lack of scalability, and limited compatibility due to vendor specific ecosystems, IEEE 802.3 Ethernet networks have the ability to overcome these limitations. Although Real-Time Ethernet (RTE) is an ongoing research domain [1, 2], current trends and activities identified TSN as a promising and widely accepted technology that may provide the capabilities to meet the requirements of industrial real-time (RT) applications and may become a platform for Industrial Ethernet [3].

Annex to ISBN 978-3-903176-28-7

### A. Problem Outline

While TSN provides different mechanisms to handle various Quality-of-Service (QoS) demands, the complexity of configuring the involved parameters in a holistic manner is a non-trivial task. While Industrial Ethernet network applications typically run by design on stand-alone networks, a challenge arises in *accommodating different Industrial Ethernet applications of various QoS requirements, as well as, best-effort traffic over a shared network*. Fig. 1 provides a sketch that shows this challenge. The SDN controller of some TSN switch topology, has to find appropriate schedules that satisfy the different hard/soft QoS guarantees required by the applications running on top. Further, given these QoS guarantees, an optimization problem arises in adaptively selecting the appropriate TSN-based scheduling algorithm. This is due to the varying properties of the scheduling algorithms, e.g., in terms of bandwidth utilization and computation time.

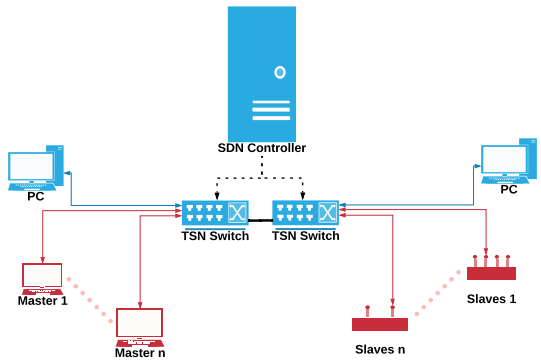


Fig. 1: The target scenario: Integrating different applications of various QoS in a TSN-based Software-defined Network.

### B. Approach & Contributions

TSN provides a viable approach to the aforementioned challenges through its Time-aware Shaper (TAS) which is capable of queuing the different application packets exhibiting varying traffic patterns such that the corresponding hard RT guarantees can be met in the presence of best-effort traffic. A refinement to the problem is providing soft guarantees

to non-RT isochronous applications. Assuming the logically centralized configuration as described by IEEE 802.1Qcc or a network-wide view provided by SDN, QoS application requirements and TSN device capabilities can be combined into calculating and distributing such schedules.

Fig. 2 shows a sketch of the approach presented in this paper that is based on the NETCONF protocol. NETCONF (Network Configuration) protocol has been developed and standardized by IETF to create a configuration management protocol to overcome the limitations of existing network configuration protocols. Newly deployed applications send the application QoS requirements and network capabilities to the SDN controller. The SDN controller, which is on the same synchronization domain as the switches, runs the scheduler software and communicates the calculated schedule variables that are required to update the Gate Control List (GCL) of the TSN switch which, in turn, controls the opening and closing of the timed-gates at the egress packet queues of the switch. Note that the schedules calculated by the SDN controller need only to be recalculated if the application QoS requirements or the network capabilities change. While we can assume that the application requirements stay mostly unchanged this approach provides the ability to account for network monitoring information, i.e., when the network capabilities change due to planned or unexpected network events.

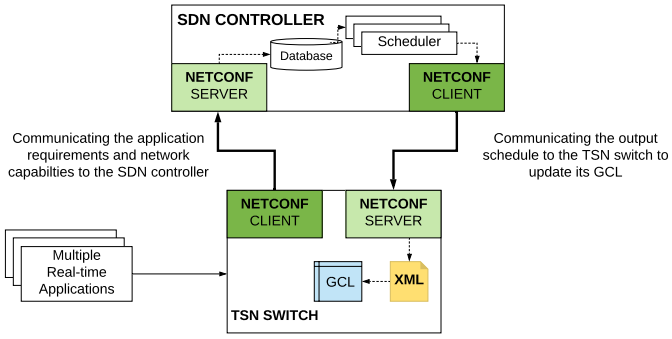


Fig. 2: NetConf-based approach to allow for bidirectional communication between SDN controller and TSN switch.

In this work, we evaluate selected schedulers for different scenarios with respect to metrics such as the bandwidth utilization, scheduler efficiency and computation time. Based on this, a decision maker is proposed for the SDN controller to select the appropriate scheduler based on the desired scenario or optimization objective. Conceptually, this corresponds to a transition [4], i.e., an adaptive selection of the used scheduler is possible upon modifications to the underlying topology.

### C. Related Work

The approach presented in [5] aims to exploit the benefits of SDN with RTE to achieve optimal routing and scheduling. Integer Linear Program (ILP) formulations were used to solve the combined problem of routing and coarse-grained scheduling of time-triggered traffic where the schedules are initialized only in end-hosts. In our approach, we rely on underlying network elements to enforce the schedules, such as the IEEE 802.1Qbv standard in TSN switches. We also assume knowledge of routing information in the considered industrial networks. The work in [6] exploits an SDN controller to provide a joint routing and scheduling solution. There, the cycle times and time-slot lengths are optimized per flow. This is fundamentally different from the scenarios considered here as the industrial Ethernet applications used are preconfigured with fixed cycle times with the goal to provides guarantees to all the present applications.

The approach in [7] investigates the computation of offline schedules by IEEE 802.1Qbv TAS over multi-hop networks. The offline schedule was calculated with respect to individual flow requirements. However, this work did not investigate the simultaneous communication of legacy devices such as industrial Ethernet applications. Another relevant approach published in [8] discusses the integration of a legacy Industrial Ethernet application into TSN to allow a topological extension and higher data rates. It only considers the accommodation of a single Industrial Ethernet system without considering simultaneous heterogeneous guarantees.

Overall, the mentioned related work does not investigate the use of TSN and SDN in accommodating multiple heterogeneous Industrial Ethernet systems over the same network. Our approach complements the previously mentioned papers in order to integrate different traffic patterns and QoS classes in an SDN-based TSN network architecture.

## II. Network and Traffic Model

Next, we consider the general hard RT isochronous traffic model, i.e., frames must be completed before their deadline while being equidistant, allowing only an ultra-low bounded jitter that is less than 1 µs and practically around 40-60 ns. We consider multi-hop layer 2 switched Ethernet networks over a full-duplex multi-speed links.

### A. Network Management Instance

The network management instance located in the SDN controller configures the traffic behavior, e.g., routing and reactive updating the forwarding tables (flow tables). The SDN controller is configured with schedulers for calculating and adapting the scheduling output variables that guarantee the isochronous communication of the hard RT applications alongside best-effort traffic applications. The computed output schedule is used to update the TSN switches' IEEE 802.1Qbv time-aware shaper's Gate Control List. Communication between the SDN controller and the TSN switches are conducted through the southbound interfaces, e.g., using NETCONF. Note that out-of-band communication is also possible as tested later on in the evaluation section.

### B. Scheduling Strategies

In the following, we consider established RT schedulers for the isochronous traffic and discuss an additional scheduling algorithm that takes advantage of the left-over unutilized link bandwidth to squeeze in applications with soft guarantees.

In RT scheduling models, each task has a release time (offset), a computation time (transmission duration) and a deadline (cycle time). Our scheduling mechanism is based on the fixed-priority scheme. In the case of RT tasks, each task has its own cycle time which also acts as a deadline, therefore the task's computation time (or in our case transmission duration) plus offset must not exceed its cycle time. We consider a static offline scheduler, i.e. all scheduling decisions are made before the execution of the system [9, 10, 11, 12]. Preemption is not allowed for express data such as isochronous traffic.

The established Rate Monotonic (RM) scheduler is appealing due to its simplicity, although it is known not to achieve full resource utilization. Since our RT industrial applications sustain time harmonic cycle times (or periods), this limitation was no longer valid [13]. In addition to RM, we also select Satisfiability Modulo Theories (SMT)-based scheduling as it showed dominance in the field of hard RT and isochronous applications [14].

After scheduling the RT applications, there may still remain unutilized link bandwidth which is offered to non-RT or best-effort traffic. In order to provide a QoS evaluation, respectively soft guarantees for that traffic class, we utilize a secondary scheduler, known as the Service Curve-based Earliest Deadline First (SCED) [15]. Similar schedulers as given, e.g., in [16] could also be considered.

*1) Rate Monotonic Scheduling:* The input requirements for RM scheduling consist of the following input parameters: the cycle time of the application frames, frame transmission duration, number of available queues and macrotick of the network (assumed to be $1\,\mu s$). These input parameters are provided to the RM algorithm for computing the following output variables: *frame offsets* and *frame priorities* where the assigned priorities are translated into their queue numbers needed to update the GCL of the TSN switch(es).

*2) SMT-based Scheduling:* SMT-based scheduling represents a powerful tool, not only in scheduling, but in any constraint-satisfaction problem. The method shows great flexibility in configuration where we obtain a schedule based on scheduling constraints that define guarantees for multiple isochronous applications. An SMT problem is a constraint-satisfaction problem, which decides whether a formula over boolean variables, formed using logical connectives, can be made true by choosing true/false values for its variables. This means that a formula $F$ is satisfiable if there exists an interpretation that makes $F$ true. When linear arithmetic is required to capture the meaning of the formulas, solvers for such formulations are called SMT solvers [17]. The SMT solver provides a model, known as the solution model, for the satisfiable context which represents one solution (out of multiple feasible solutions) for the given problem. The constraints used to populate our model in the SMT solver are explained in detail in Section II-C.

The input to the SMT solver comprises: the maximum allowed end-to-end latency, the period/cycle time of the frame, the frame data size, the network precision which is used as a reference for the macrotick of the network, the propagation delay on the medium, the number of available queues in the switch, the link speed and the frame transmission duration. These constants, alongside the sought variables: *frame offsets* and the *queue number assigned to the frame*, are provided as part of the constraints. The SMT solver iterates the values of the variables, which are the frame offsets and queue numbers, until it finds a valid solution which satisfies all constraints concurrently. Note that this only represents one solution out of many possible solutions. The calculated frame offsets, frame transmission durations and queue number assignments translate to *gate opening and closing events*.

*3) Service Curve-based Earliest Deadline First:* To utilize the remaining bandwidth that is leftover by the RT applications we use an approach that is known as SCED [15]. An example of applications that benefit from this unused bandwidth is Controller-to-Controller (C2C) communication, which is non-isochronous RT traffic. C2C traffic comprises small frames that are sent between the controllers of the Industrial Ethernet systems e.g. to exchange non-RT process parameters and data. This traffic is considered to have much larger cycle times than hard RT traffic. It may hence be prioritized over best-effort traffic but not over the isochronous traffic.

Service curves [18] were first introduced to provide a broad abstraction for the characterization of the service provided by network elements. This enabled the calculation of QoS guarantees, such as backlog and delay, for data flows given a characterization of their respective traffic burstiness and the elements' service curves [19]. SCED was proposed as it achieved the largest possible schedulability when provided with a set of requirements on the delay and a set of specifications describing the burstiness of the data flow [15]. In the considered scenario this is obtained from characterizing the C2C communication.

To allocate resources given some delay and throughput requirements, conditions have to be met to allow a network node to simultaneously guarantee all the service curves for all connections. Basically such a condition is known as the feasibility test for service curve allocations which states that in order to satisfy all constraints, the sum of the calculated service curves of these connections has to be less than or equal to the capacity curve (cumulative capacity per time-slot of the system). The capacity curve, in our case, is shaped according to the RT schedule we compute beforehand. Upon satisfying the feasibility test we implement the SCED policy to the given service curves that represent different C2C data flows. SCED states that each packet is assigned a deadline and packets are served in an earliest deadline first fashion. The developed SDN controller architecture is depicted in Fig. 3 showing the interaction of the different controller modules.

### C. SMT Scheduling Constraints

In the case of SMT-based schedules we require scheduling constraints in order to guarantee the isochronous traffic patterns. These constraints comprise the frame offset variables and flow queue variables per application. This is required to compute a schedule for the timed gates of the TSN switch(es).

We build on the constraints from [7], modify them according to our needs and add additional constraints to fit our model and approach. Note that we took into consideration each individual frame such that we can apply changes to the frames separately within each application flow. This is a major difference to the approach in [7] as they only considered the first frame of each application. The formulas are not provided here due to space restriction.

*1) Frame Constraint:* The frame offset of any frame scheduled in the network has to be greater than or equal to the start of the frame period (or cycle time). This ensures that the frame offset cannot take either a negative number or lie within a period of another precedence frame. Additionally, the entire transmission window (offset plus frame duration) has to fit within the frame period to ensure that the frame would not miss its hard deadline.

*2) Link Constraint:* A crucial constraint in TSN is to ensure that no two frames can overlap in the time domain. We achieve this by comparing all the frames individually between each two applications.

*3) Jitter Constraint:* The jitter constraint allows the shifting of consecutive frames of the same application by a maximum jitter value to allow flexibility in assigning schedules. This means that the consecutive frames of the same application could possess some jitter to find satisfiable schedules when no satisfiable schedules can be computed with strictly equidistant frames. Regarding the hard real-time applications used, we need the jitter of the applications to be within the 1 µs window.

*4) Flow Transmission and End-to-End Constraint:* The flow transmission constraint states that the propagation of a flow must follow the sequential order along the routed path of the flow. The constraint specifies the maximum end-to-end latency between the arrival time of a frame of a flow and the sending time of that frame at the source.

*5) Queue Constraint:* This ensures that each RT application is placed in a separate queue to guarantee isochronous communication for all the applications simultaneously.

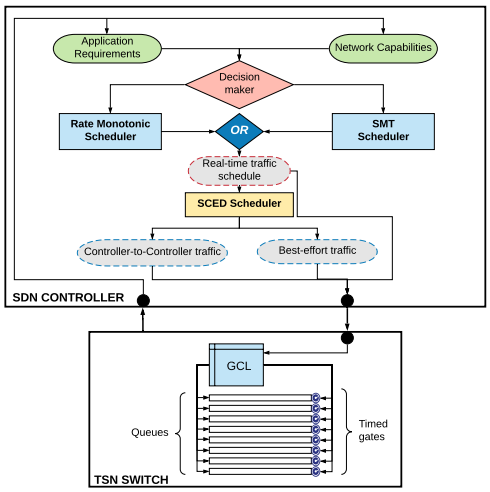


Fig. 3: SDN controller architecture including TSN components.

## III. Evaluation Environment

We conduct experiments to benchmark and compare the scheduler performance where we used a linear topology of two TSN switches before validating our results in a simulation environment. Note that the schedules are calculated and applied for each switch individually.

### A. Experiment and Simulation Settings

We use the `OpenDaylight` Controller which possesses a model-driven SDN controller that utilizes YANG models and XML-based files to support its features [20].We use the Z3 v4.4.1 solver [21], as SMT solver, which contains algorithms for solving linear optimization objectives in addition to the usual SMT functionalities. The scheduling problem constraints are defined and populated using the notations of the Z3 solver. The generated output of the scheduling algorithms, given in an XML-format file, is communicated to the TSN switch to update its GCL. Simulations of the performance scaling with an increasing path length of the linear topology were performed using the discrete event-based network simulator OMNEST which is the commercial version of OMNeT++. There we use a framework [22] for including the functions of TSN, especially IEEE 802.1Qbv Time-Aware Shapers.

### B. Industrial Applications and Scenarios

We consider RT applications that represent different setups of two Industrial Ethernet technologies (Sercos III and EtherCAT) where the cycle times are ultra-low, i.e. $\leq 500\,\mu s$. A list of these RT applications is shown in Table I spanning digital and analog (I/O) applications, Servo-axes and distributed image processing. The considered C2C communication is non-isochronous RT traffic, having higher cycle times (1-10 ms) and 100 bytes of data per cycle. Best-effort traffic is considered as a non-deterministic traffic which can be preempted.

For the evaluation we define scenarios based on a combination of the applications from Table I. In the first scenario (*packing problem*) we iterate over all mentioned hard RT applications until no more applications can be added without violating the QoS guarantees. The second scenario is a validation problem, where we validate the different schedulers (SMT, RM and SCED) against different sets of applications.

## IV. Evaluation Results

One objective of our evaluation is to identify which scheduler is more suited to use in which networking scenario. The main evaluation metrics used are the utilized bandwidth and the computation time of the schedulers. The scheduler efficiency is measured as the number of satisfied RT applications supported by a common schedule. Note that an RT schedule is considered unsatisfiable in SMT when the solver cannot reach a solution model for the specified applications. However, for RM, an RT schedule is considered unsatisfiable when either preemption of frames occur or deadlines are missed. The data rate used for sending frames between TSN switches on the linear topology is 1 Gbps for the following experiment.

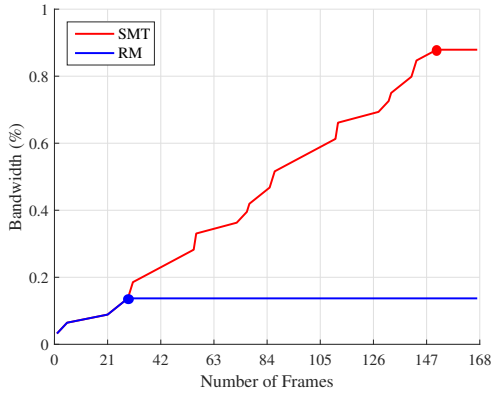


Fig. 4: Bandwidth utilization varying over frames (of applications) added, using SMT and RM schedulers. The *blue dot* shows the last reading before the RM scheduler was preempted and the *red dot* shows the last reading before the SMT scheduler showed an unsatisfiable schedule.

*1) Packing Problem:* For our first case, we test the SMT-based and RM scheduling by increasing the number of applications until the scheduling algorithms no longer find a *satisfiable schedule*. We iterate sequentially over all the hard RT applications with cycle times $\leq 500\,\mu s$.

Figure 4 shows the achieved bandwidth utilization per scheduler before not being able to provide a satisfiable schedule. We can observe that the SMT scheduling algorithm can allocate many more frames, corresponding to RT applications, before having an unsatisfiable schedule, as compared to the RM scheduling algorithm. The reason behind the RM scheduler poor application allocation performance is that frames are preempted at an early stage and are scheduled consecutively. Since the smallest cycle time application is always scheduled first, the sum of all the frame transmission duration of all the consecutive RT applications has to be less than or equal to the difference between the cycle time and the frame transmission duration of the smallest cycle time application. The SMT scheduler has the flexibility in assigning frames to time-slots as long as the specified constraints are all met. Hence, in the case of SMT, the frame transmission duration of all the frames has to individually be less than or equal to the difference between the cycle time and the frame transmission duration of the smallest cycle time application.

In a second experiment we measure the computation time needed by each scheduler when adding up to 168 new frames (Figure not shown for space constraints). We run our scheduler for 20 iterations for every additional application and calculate the average computation time alongside corresponding confidence intervals. We observe that the RM scheduler sustains a steady and low computation time ($\sim$ 60 ms), regardless of the number of applications/frames added to the scheduler. While the SMT computation time is comparable to RM for a low number of frames ( 60ms for $\sim$ 10 frames) it increases later by roughly three orders of magnitude ($\geq$ 30 sec for $\sim$ 168 frames). The reason behind the RM scheduler behaviour is that it always iterates over the same hyper period, regardless of the number of applications. The SMT scheduler requires however longer computation times because additional replicas of the constraints populate the SMT solver per added application.

*2) Scheduler Validation:* Next, we consider the SMT scheduler change in efficiency under non-isochronous and isochronous hard RT traffic. We iterate over a number of RT applications, first subject to the jitter constraint, which guarantees isochronous traffic, and then after removing this constraint to observe how many more RT applications of the same type can be satisfied when they exhibit non-isochronous traffic patterns. We iterate over the applications until no satisfiable schedule can be computed. In Fig. 5 we observe a steep increase in bandwidth utilization in the number of frames that are added over a SMT solver configured without the jitter/isochronous constraint ($\sim 90\%$ utilization) in comparison to the one configured with the jitter constraint ($\sim 15\%$ utilization). This is because dropping the jitter constraint increases the flexibility of the SMT solver substantially as it can assign the frames in any time-slot inside its corresponding cycle time.

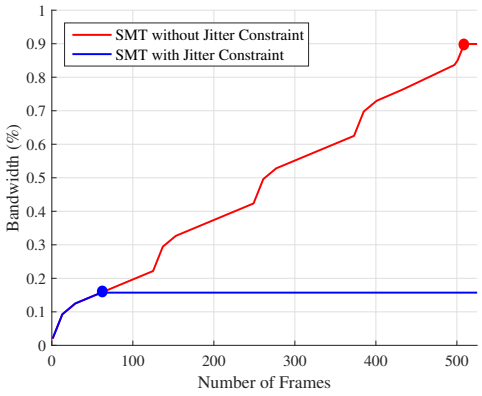


Fig. 5: Bandwidth utilization as function of the number of frames that are added over a SMT solver configured with and without the jitter/isochronous constraint. The blue dot and the red dot show the last value before the SMT scheduler returns an unsatisfiable schedule.

*3) Linear Topology Evaluation:* Next, we briefly report evaluation results when increasing the number of hops in a line topology under RM scheduling where all links possess a bandwidth of 1Gbps. We observe an increase in the round-trip time which for three fresh applications with cycle times $\{250, 125, 62.5\}\mu s$ and data sizes $\{1488, 600, 228\}$ Bytes, respectively. The frames observe queueing delays of the schedules at each hop in addition to store and forward delays on the forward and backward paths. We note that the delay increase is not necessarily linear in the number of hops as it includes store and forward delays of frames of different sizes at each hop. This may be particularly harmful for applications with strict jitter constraints.

*4) SCED Validation:* Finally, we investigate the effect of increasing the number of RT applications on the maximum number of non-real time C2C communication applications that can be supported. Note that we do not compare the perfor-

TABLE I: Application profiles: EtherCAT (ET) and Sercos III (SIII).

| Application Label | Application Description | Cycle Time ($\mu$s) | Data Size (bytes) |
|---|---|---|---|
| A | ET | 31.25 | 125 |
| B | ET | 125 | 512 |
| C | ET | 500 | 1500 |
| D | ET | 62.5 | 400 |
| E | ET | 250 | 1500 |
| F | SIII | 62.5 | 325 |
| G | SIII | 31.25 | 248 |
| H | SIII | 500 | 3000 |
| I | SIII | 2000 | 5000 |
| J | SIII | 1000 | 6000 |
| K | SIII | 2000 | 12000 |

mance of the RT schedulers (SMT and RM) here but rather validate the use of the SCED scheduler which is built on top of the computed RT schedule. We iterate over the applications mentioned in Table I in the following order: applications I, J, K, C and H, until we can no longer compute a satisfiable schedule. Given the RT schedule, we find feasible service-curve allocations by computing the maximum number of C2C connections that can be supported over this RT schedule. We observe how the maximum number of supported C2C connections diminishes as more RT applications are added to our schedule (Fig. not shown for space reasons).

## V. Conclusion and Outlook

In this work we addressed the need for a unified network supporting multiple legacy devices, such as current Industrial Ethernet systems, alongside other traffic classes, such as controller-to-controller traffic and best-effort traffic. This proved to be a vital issue for the realization of integrated industrial IoT, where heterogeneous cyber-physical systems communicate control and data information over a shared network. We have made this possible by utilizing TSN using its enhanced scheduling standard IEEE 802.1Qbv, to calculate and implement schedules using a Software-defined Networking controller to simultaneously provide real-time guarantees to all considered traffic classes.

We evaluated real-time schedulers in different networking scenarios and discussed the design of a decision maker functionality in the SDN controller that ensures the selection of the better scheduling mechanism per given scenario. We further considered scheduling a mixture of traffic of different requirements, such as non-isochronous real-time traffic with controller-to-controller traffic that requires a soft real-time guarantee. This allowed a realization in the SDN controller comprising multiple levels of scheduling mechanisms per traffic class for efficient bandwidth utilization. In future we aim to analyze more Industrial Ethernet systems, such as PROFINET, POWERLINK, EtherNet/IP, etc, taking into consideration their different traffic patterns.

## References


[1] R. Schlesinger, A. Springer, and T. Sauter, "Automatic packing mechanism for simplification of the scheduling in profinet irt," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 5, pp. 1822–1831, Oct 2016.

[2] M. Knezic, B. Dokic, and Z. Ivanovic, "Theoretical and experimental evaluation of ethernet powerlink pollresponse chaining mechanism," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 2, pp. 923–933, April 2017.

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

[4] B. Alt et. al., "Transitions: A Protocol-Independent View of the Future Internet," *Proceedings of the IEEE*, vol. 107, no. 4, pp. 835–846, 2019.

[5] N. G. Nayak, F. Dürr, and K. Rothermel, "Time-sensitive software-defined networks for real-time applications," Technical Report Computer Science 2016/03, University of Stuttgart, Germany, Tech. Rep., 2016.

[6] E. Schweissguth, P. Danielis, C. Niemann, and D. Timmermann, "Application-aware industrial ethernet based on an SDN-supported TDMA approach," in *Proceedings of IEEE World Conference on Factory Communication Systems (WFCS)*, May 2016, pp. 1–8.

[7] S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner, "Scheduling Real-Time Communication in IEEE 802.1Qbv Time Sensitive Networks," in *Proceedings of ACM International Conference on Real-Time Networks and Systems*, ser. RTNS, 2016, pp. 183–192.

[8] S. Nsaibi, L. Leurs, and H. D. Schotten, "Formal and simulation-based timing analysis of Industrial-Ethernet sercos III over TSN," in *Proceedings of IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, Oct. 2017, pp. 1–8.

[9] A. Burns, "Scheduling hard real-time systems: a review," *Software Engineering Journal*, vol. 6, pp. 116–128(12), May 1991.

[10] L. Sha, K.-E. Abdelzaher, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, and A. K. Mok, "Real time scheduling theory: A historical perspective," *Real-Time Syst.*, vol. 28, no. 2-3, pp. 101–155, Nov. 2004.

[11] S.-H. Oh and S.-M. Yang, "A modified least-laxity-first scheduling algorithm for real-time tasks," in *Proceedings Fifth International Conference on Real-Time Computing Systems and Applications*, 1998, pp. 31–36.

[12] Y.-S. Yen, W. Chen, J.-C. Zhhuang, and H.-C. Chao, "Sliding weighted fair queueing scheme for real-time applications," *IEE Proceedings - Communications*, vol. 152, pp. 320–326(6), June 2005.

[13] V. Shinde and S. C. Biday, "Comparison of real time task scheduling algorithms," *International Journal of Computer Applications*, vol. 158, no. 6, pp. 37–41, 2017.

[14] W. Steiner, "An Evaluation of SMT-Based Schedule Synthesis for Time-Triggered Multi-hop Networks," in *Proceedings of IEEE Real-Time Systems Symposium (RTSS)*, 11 2010, pp. 375–384.

[15] H. Sariowan, R. L. Cruz, and G. C. Polyzos, "SCED: A generalized scheduling policy for guaranteeing quality-of-service," *IEEE/ACM Transactions on Networking*, vol. 7, no. 5, pp. 669–684, Oct 1999.

[16] L. Zhao, P. Pop, Z. Zheng, and Q. Li, "Timing analysis of avb traffic in tsn networks using network calculus," in *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, April 2018, pp. 25–36.

[17] L. De Moura and N. Bjørner, "Satisfiability modulo theories: Introduction and applications," *Commun. ACM*, vol. 54, no. 9, pp. 69–77, Sep. 2011.

[18] R. L. Cruz, "Quality of service guarantees in virtual circuit switched networks," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 6, pp. 1048–1056, Aug 1995.

[19] H. Sariowan, R. L. Cruz, and G. C. Polyzos, "Scheduling for quality of service guarantees via service curves," in *Proceedings of International Conference on Computer Communications and Networks - IC3N'95*, Sep. 1995, pp. 512–520.

[20] J. Medved, R. Varga, A. Tkacik, and K. Gray, "Opendaylight: Towards a model-driven sdn controller architecture," in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, June 2014, pp. 1–6.

[21] B. N. de Moura L., *Z3: An Efficient SMT Solver*, ser. Lecture Notes in Computer Science. Springer, 2008, vol. 4963.

[22] *NeSTiNg - Network Simulator for Time-sensitive Networking*, University of Stuttgart, last Updated: July 2018. [Online]. Available: https://gitlab.com/ipvs/nesting