Survey on Scheduling Approaches in TSN

Leon Kist, Philippe Buschmann*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: leon.kist@tum.de, phil.buschmann@tum.de

***Abstract*—Modern concepts such as Industry 4.0 or in-vehicular networks feature complex, interconnected systems. These systems often depend on highly time-critical communication. One approach to provide this type of communication is supplied by Time-Sensitive Networking (TSN). TSN is standardized in IEEE 802.1Q and encapsulates several different mechanisms. Among these mechanisms is the Time-Aware Shaper (TAS), which allows the creation of Time Division Multiple Access (TDMA) schemes for traffic in the network. However, the synthesis of these schemes entails several problems, such as exponential computation times. Another difficulty arises from the prioritization of traffic, i.e., which streams to prioritize in order to maximize the functionality of applications.**

In this paper we survey two different approaches to scheduling in TSN. The first approach leverages hierarchical structures in factory automation networks to simplify schedule creation. The second approach aims at improving the Quality of Service (QoS) of less time-critical traffic, while still adhering to the requirements of more time-critical traffic. Both approaches succeed in their goals.

***Index Terms*—time-sensitive networking, scheduling**

1. Introduction

Modern concepts such as Industry 4.0, smart factories, and in-vehicular networks feature complex, interconnected systems which often depend on various IoT devices and sensors to enhance productivity and avoid errors or malfunctions [1] [2]. In order to provide such functionality, communication between different elements of these systems, needs to adhere to strict timing constraints or even has to have real-time properties [3].

One approach to meet these constraints is provided by *Time-Sensitive Networking* (TSN). TSN uses wired ethernet and is standardized by the IEEE TSN task group, which has released several standards for TSN starting in 2012. One such standard, IEEE 802.1Qbv (now included in IEEE 802.1Q), introduces the *Time-Aware Shaper* (TAS). The TAS can be used to create a *Time Division Multiple Access* (TDMA) scheme for traffic in the network, based on a given schedule [4] [5] [3] [6].

Schedule creation for the Time-Aware Shaper however, entails a number of different problems. The creation of schedules itself can be considered as a NP-Hard problem. The computational effort associated with creating schedules is therefore often large. This is especially true in many common applications where networks are large and many streams of data are exchanged. Another issue is the prioritization of different streams in the network. Some components can tolerate missing communication deadlines, while others would fail in the presence of a single deadline-miss. In order to maximize the functionality of applications it is necessary to avoid deadline-misses for highly critical traffic, while still providing adequate properties for less critical streams [3] [6].

The goal of this work is to survey different approaches or improvements to scheduling for the *Time-Aware Shaper* in TSN. To do this we will first explore some related work in Chapter 2. Afterwards in Chapter 3 some essential background information will be explained. Next, we will analyze the scheduling approaches in Chapter 4. After this in Chapter 5 we will compare the approaches. Then finally Chapter 6 will provide some conclusions and a possible outlook for future work.

2. Related work

As this is a survey of different, preexisting works, we will explore some of the mentioned works in more depth in Chapter 5. For now, we only give a short outline of the main papers we are surveying.

In the first work we analyze Hellmanns et al. [3] describe an hierarchical scheduling approach that exploits common properties of Factory Automation Networks.

Second we explore a paper by Houtan et al. [6], which seeks to improve existing scheduling approaches in regards to the performance of less time critical best effort (BE) traffic.

While there exist numerous other papers on scheduling approaches or improvements thereof, we are not able to analyze them further, due to the limited scope of this work. Other work includes an approach by Oliver et al. [7], which focuses on the creation of schedules using the first-order theory of arrays. Another paper by Vlk et al. [8], seeks to improve the performance of scheduled traffic, as well as the creation of schedules by improving hardware and the scheduling mechanism itself. A paper by Syed et al., explores a Mixed-integer Programming based approach for both scheduling and routing combined [9].

3. Background

In this chapter we provide some background information necessary for understanding the surveyed approaches.

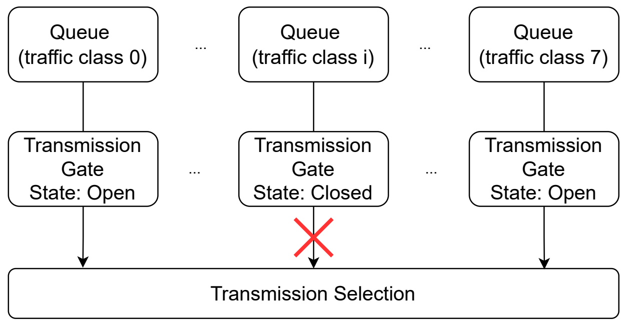


Figure 1: Simplified visualization of transmission selection (based on Figure 8-14 of IEEE 802.1Q) [4]

3.1. The Time-Aware Shaper

As already mentioned, the TAS can be used to create TDMA-schemes for traffic in a network. To achieve this, IEEE standard 802.1Qbv specifies a gating procedure to restrict the amount of traffic forwardable by a switch at each point in time [3] [6].

To understand this mechanism, it is necessary to first explore transmission selection in TSN capable switches (as specified by IEEE 802.1Q). Such switches possess between 1 and 8 separate queues for each of its egress ports (transmission selection happens separately for each egress port). Each queue corresponds to a traffic class. Traffic classes are categorizations assigned to frames/streams based on their priority. Frames are chosen for transmission based on traffic class and transmission selection algorithms supported by the queues [4].

The gating mechanism introduced by IEEE 802.1Qbv now specifies an additional *Transmission Gate* for each queue. This gate can either be *Open* or *Closed*. If the *Transmission Gate* for a queue is closed, then frames of that queue are ineligible for transmission selection. A simplified version of this is illustrated in Figure 1.

The state for each gate is controlled by the so called *Gate Control List* (GCL). The GCL is an ordered list of *Gate Operations*. Each *Gate Operation* consists of a vector specifying the state for all gates and a time interval specifying how long the state should persist. The sum of all such time intervals is referred to as *network cycle*. All egress ports cycle over a separate GCL. Schedule synthesis effectively corresponds to the creation of *Gate Control Lists*. Suitable timing allows not only for scheduling on traffic class level, but also on stream-level [5] [3].

3.2. Schedule Creation

Approaches for schedule creation can generally be distinguished into two distinct groups: *exact approaches* and *heuristic approaches Exact approaches* calculate solutions with provable properties, such as optimality or satisfiability. These approaches are often based on mathematical methods, like *Integer Linear Programming*, *Satisfiability Modulo Theories* (SMT), or *Optimization Modulo Theories* (OMT) [3] [6].

While the calculation of exact results can be advantageous, it is generally a NP-Hard problem. Using mathematical methods can therefore cause issues, such as very high resource consumption or exponentially long computation times. This is especially problematic when calculating schedules for larger networks [3] [6].

To avoid these problems, it is also possible to use *heuristic approaches*. Heuristic approaches forgo the calculation of provably optimal solutions in favor of an optimized calculation duration [3] [6].

4. Analysis

In this chapter we analyze the previously mentioned surveyed work. First, we explore the hierarchical approach proposed by Hellmans et. al [3]. After this we dive into improvements to the performance of Best-effort (BE) traffic introduced by Houtan et al. [6].

4.1. Hierarchical Approach

As already noted, the creation of schedules can be computationally intensive. This is especially true for large networks or networks where a large amount of different data-flows are exchanged. In order to reduce this overhead, Hellmans et al. propose a scheduling approach leveraging the hierarchical structure of Factory Automation Networks [3].

4.1.1. Terminology and Network Characteristics. Before we elaborate on the actual scheduling approach proposed in [3], we need to explain some specific terminology and assumptions about the network.

The first term we define is *stream isolation*. This refers to scheduling where each stream is assigned a specific (separate) time slot in the created TDMA-scheme. In contrast, *stream batching* refers to scheduling where multiple streams share a single time slot.

Next we briefly explain certain characteristics of *factory automation networks* as used in e.g., smart factories [10]. Such networks are hierarchical in nature and can therefore be divided into several sub-networks. These sub-networks are connected by *boundary switches*.

Based on [11] and [3] we differentiate between several types of traffic, each with unique deadline/latency requirements. However, to understand this work it is mostly important to understand that such types exist. The only type we need to elaborate on in a more detailed manner is *isochronous traffic*. This type of traffic has strict deadline requirements and occurs periodically. The exact timing of when an isochronous stream is sent is chosen during design time. All other traffic types have less strict timing requirements.

We can distinguish two types of streams of isochronous traffic: *intra-level streams* (within a sub-network) and *inter-level streams* (between sub-networks). We assume that *intra-level streams* have even stricter timing constraints than *inter-level streams* [3].

4.1.2. Approach. Now that the necessary terminology and network characteristics have been explained, we can explore the scheduling approach. This approach envelopes two separate parts: (1) a *phase model* which isolates the different types of traffic in the network and (2) a *schedule creation* mechanism for isochronous traffic [3].

The **phase model** essentially divides the *network cycle* into separate *cycle phases*. Each *cycle phase* is assigned to a traffic type. This *cycle phase* is then exclusively reserved to this single traffic type. Scheduling happens separately for each *cycle phase*. This simplifies scheduling, as schedulers only ever have to consider one type of traffic at a time. It also allows the usage of different scheduling mechanisms for different types of traffic [3].

The **schedule creation** mechanism for isochronous traffic consists of two stages executed in the following order: 1) intra-level scheduling (intra LS) and 2) inter-level scheduling (inter LS). Separating the traffic in such a manner is possible due to the hierarchical nature of factory automation networks, as well as the differences in timing constraints between intra-level streams and inter-level streams. Both phases are visualized in Figure 2.

Stage 1): *Intra LS* creates a separate schedule for each *machine ring* sub-network. This drastically reduces the complexity of schedule synthesis. Schedules incorporate both intra-level streams and inter-level streams. However, inter-level streams are only forwarded to their respective boundary router (on the local ring) in this stage. There they are queued until the second stage. *Intra LS* uses *stream-isolation* to accommodate the strict timing requirements of intra-level streams. Any of the previously mentioned schedule creation approaches (e.g., heuristic, SMT, OMT) can be used in this stage.

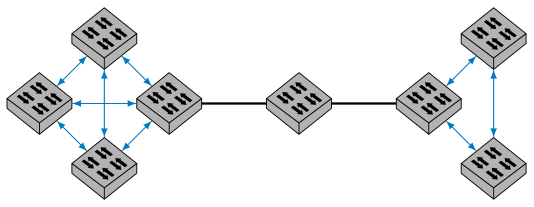
Stage 2): *Inter LS* is used to create schedules for inter-level streams (which are buffered at their boundary routers after the first stage). However, in contrast to stage 1), *stream-batching* is used instead of *stream-isolation*. All streams queued at a boundary router after stage 1) are grouped together. Schedule creation itself does not use any of the previously mentioned approaches. Instead, the forwarding of all isochronous traffic is simulated, as if there was no scheduling (i.e., all gates for isochronous traffic are open). The simulation then determines for how long each gate has to be open. After all gates are closed the *cycle phase* for isochronous traffic can end. As there is no other traffic in the network during stage 2) (due to the *phase model*), the simulation should be able to accurately predict forwarding in the network.

It is important to note, that should stage 2) detect a deadline failure, it has few mechanisms to prevent this. The only possible recourse is to try and manipulate the arrival time of said stream at the boundary switch in stage 1) to gain an advantageous position in the queue of the switch. However, as it is assumed that inter-level streams are less time-critical, no detailed strategy is presented [3].

4.2. QoS Improvements

As already mentioned, TSN can be used to guarantee the timing requirements of time-critical traffic in networks. However most networks also feature less (or non) time-critical traffic. While delaying this traffic might not be as detrimental to the application, as delaying highly time-critical traffic, it can still affect its performance negatively. This approach therefore aims to improve the Quality-of-Service (QoS) of less time-critical BE traffic, while still guaranteeing the deadlines of time-critical streams [6].

4.2.1. Terminology and OMT. Before we can explore the actual approach, we once again need to define some terminology.


(a) Phase 1): *Intra LS*

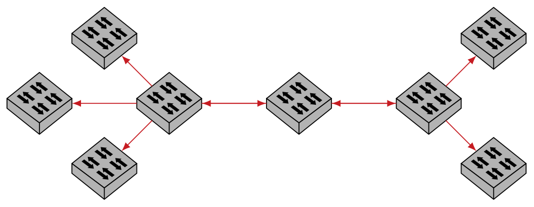

(b) Phase 2): *Inter LS*

Figure 2: Visualization of phases in the heuristic approach for a simple network

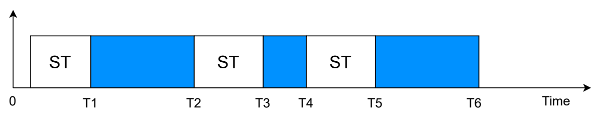


Figure 3: Visualization of slack (in blue) based on [6]

As in the previous approach we need to differentiate between multiple types of traffic in the network. However, for this approach we only need to distinguish highly time-critical scheduled traffic (ST) and less time-critical BE traffic.

This approach uses the previously mentioned OMT, and can therefore be classified as an exact approach. OMT allows the scheduling problem to be expressed as a constrained optimization problem. This problem can then be solved using SMT/OMT solvers [6]. The approach seeks to improve a previous approach by Craciunas et al. proposed in [12].

4.2.2. Approach. To accommodate BE frames, the concept of *slack* is introduced. *Slack* describes a time interval after a ST frame is sent in which no other ST frame can be sent. This time may then be used for BE frames. This is visualized in Figure 3 This approach essentially aims to find feasible schedules for ST traffic, while still guaranteeing a certain *slack*. In order to achieve this, several new constraints and objective functions are introduced. As the scope of this work is quite limited, we will only explain their objective and effects. For a more detailed view, we refer to [6].

To optimize QoS of BE traffic, the approach introduces four new **constraints**.

The *porous link constraint* is a modification of the *link constraint* introduced in [12]. The original constraint had the purpose of preventing temporal overlap of frames on the same link, i.e., two frames being sent at the same time. This modified version also incorporates *slack*, it therefore disallows overlap of ST frames with previous ST frames and their associated slack.

The *slack size constraint* has the purpose of defining the acceptable *slack* size for each frame.

The *hop slacks constraint* encompasses several equations, which ensure that the total amount of *slack* on a link is within acceptable bounds.

The *equal slack constraint* is optional. Its purpose is to enforce equal slack sizes for all frames on one link. This can be used to improve the optimization time, as it limits the amount of eligible ST schedules [6].

In addition to these new constraints, three new **objective functions** are introduced.

In previous work a *minimization* objective function is used. This function minimizes the total offset of ST streams. Therefore, schedules created using this function (subject to the preexisting constraints mentioned), feature all ST traffic right in its beginning. This can be detrimental to BE traffic in the network.

The first objective function introduced is the *maximization* function. In contrast to the *minimization* function, the *maximization* function, puts all ST frame transmissions as close to their deadlines as possible. All deadlines are therefore still guaranteed, however BE traffic can (depending on the amount of ST frames) be scheduled much earlier (in many cases). This function does not use the notion of *slack* and is therefore still only subject to the preexisting constraints.

The *sparse schedule* objective function aims to maximize the total amount of slack on each link. Schedules created by this function distribute ST frames sparsely, with gaps (*slack*) in between for BE traffic. This function is subject to both the preexisting and newly introduced constraints (without the *equal slack constraint*).

The *evenly sparse schedule* objective function is essentially the same as the *sparse schedule* objective function, however it is also subject to the *equal slack constraint*. This leads to the creation of schedules with an even distribution of ST frames. Gaps (*slacks*) between ST frames are of the same length.

Using the *sparse schedule* function or the *evenly sparse schedule* function has the advantage of avoiding long queueing times for BE frames, as well as reducing the search space, which can lead to faster computation times [6].

5. Evaluation

This chapter is dedicated to the evaluation of the introduced approaches. First, we discuss the results achieved by the two approaches. Afterwards we assess their compatibility.

5.1. Results

In this section we present and compare the results measured in both papers.

5.1.1. Hierarchical Approach. To evaluate the effectiveness of their approach, Hellmans et al. compare the performance of their approach to that of a (heuristic) global 1-stage approach. The same approach is used for *intra LS* in the hierarchical approach. The following metrics are evaluated: execution time, cycle phase duration, and the average number of GCL entries per port. Each metric is evaluated for a variety of topologies using different amounts of sub-networks and sub-network sizes. The amount of streams in the network is varied as well.

The impact of different topologies is minor compared to that of the number of streams in the network. Regarding *cycle phase length*, both approaches perform about the same with slightly shorter (<0.1ms) cycle phases using the global approach. The *amount of GCL entries* is comparable for both approaches when fewer streams are in the network ($\leq$500). When more streams are present, the hierarchical approach significantly outperforms the global approach, up to a factor of four. This is because the global approach requires two entries per hop, per stream. The hierarchical approach acts similarly in stage 1), but only requires two additional entries in stage 2). The largest difference occurs when comparing the *execution times*. Similarly to the GCL lengths, the execution times are comparable for fewer streams ($\leq$100), with the global approach even outperforming the hierarchical approach. This is due to the overhead caused by the increased number of calls to the scheduler in stage 1. However, for more streams the global approach is outperformed up to a factor of more than 100. For more than 1250 streams, the global approach is unable to find a solution (even after more than 200h), due to insufficient hardware capabilities [3].

5.1.2. QoS Improvements. To evaluate their results, Houtan et al. simulate a network with six end stations, two connected TSN switches and ten streams. Streams have varying probabilities to be ST or BE. To measure the QoS of BE streams, the end-to-end delay of BE streams, deadline misses (for BE streams), as well as execution time (of schedule synthesis) were chosen.

Regarding end-to-end delay, the *minimization* function performs the worst out of all four functions. The *sparse schedule* function and *evenly sparse schedule* function perform about the same. The *maximization* function performs the best out of all objective functions. This remains true across all scenarios. The *maximization* function outperforms the other functions, since it schedules ST traffic as late as possible. This allows BE traffic to be scheduled earlier, thereby decreasing its end-to-end delay. Weaknesses of the *maximization* function reveal themselves when looking at deadline misses. Here the *maximization* function, as well as the *minimization* function produced significant amounts of deadline misses for one of the tested scenarios. Both the *sparse schedule* function and *evenly sparse schedule* function produce virtually no deadline misses in any of the scenarios. The *sparse schedule* function and *evenly sparse schedule* function also perform the best regarding execution time. The *minimization* function performs significantly worse than either of the previously mentioned functions. The performance of the *maximization* function varies depending on the amount of ST streams in the network. The more ST streams are present, the worse its performance becomes. In the presence of large amounts of ST streams, the *maximization* function is significantly outperformed by the *minimization* function [6].

5.1.3. Comparison. Both approaches have quite different objectives and mechanisms. This, in conjunction with the vastly different network topologies used for their measurements, make a comparison between the results of the approaches quite difficult. The only metric measured for both approaches is execution time. Both approaches succeed in significantly improving the schedule creation time, under the right circumstances. For the heuristic approach, these improvements depend on the amount of streams in the network. Only in the presence of more than 100 streams, does the hierarchical approach provide significant improvements. However, the more streams are present, the larger the improvements. The global approach in turn is only measured for a fairly small network with few streams. However here it manages to significantly improve the execution time [3] [6].

We can therefore conclude that for small networks or networks with few streams (≤ 100) the global approach would yield larger performance improvements. For networks with a great number of streams (>100) however, the hierarchical approach is likely to achieve greater improvements. In networks with more than 1250 streams it is possible that a hierarchical approach is the only feasible solution (depending on the existing hardware).

5.2. Compatibility

The first scheduling approach introduced divides the *network cycle* into several different phases for each type of traffic. Scheduling is done for each type of traffic separately. In contrast, the second approach aims at finding a unified schedule for all traffic combined. As the first approach requires separation of traffic types and the second approach requires knowledge about all traffic, a direct combination of both approaches seems infeasible. However the notion of leveraging hierarchical structures in the network itself and using a multi-stage approach could be used for multiple types of traffic combined. In a setup like this, something akin to the second approach might be applicable.

6. Conclusion and Future Work

In this survey we look at two very different approaches to improve scheduling for TSN. The first approach we survey aims to utilize hierarchical characteristics of the network to improve scheduling. The second approach tries to improve the performance of less time-critical traffic by better utilizing the deadlines of more critical traffic. Both approaches succeed in improving different aspects of scheduling.

Looking at the hierarchical approach, we are able to see that leveraging characteristics of certain types of networks allows for significant improvements to scheduling. Determining and utilizing such characteristics could provide an excellent avenue for future work.

Another interesting approach might be to combine the introduced hierarchical, multi-stage approach with other approaches in order to explore their compatibility.

From the improvements to QoS of less time-critical traffic we can see that synthesizing feasible schedules for highly time-critical traffic is possible, even in the presence of other objectives. It might therefore also be worth investigating if there are issues with other scheduling approaches which can be mitigated or solved like this.

References

[1] "What is Industry 4.0?" https://www.ibm.com/topics/industry-4-0, [Online; accessed 21-March-2022].

[2] W. Zhou and Z. Li, "Implementation and Evaluation of SMT-based Real-time Communication Scheduling for IEEE 802.1Qbv in Next-generation in-vehicle network," in *2020 2nd International Conference on Information Technology and Computer Application (ITCA)*, 2020, pp. 457–461.

[3] D. Hellmanns, A. Glavackij, J. Falk, R. Hummen, S. Kehrer, and F. Dürr, "Scaling TSN Scheduling for Factory Automation Networks," in *2020 16th IEEE International Conference on Factory Communication Systems (WFCS)*, 2020, pp. 1–8.

[4] "IEEE Standard for Local and Metropolitan Area Network–Bridges and Bridged Networks," *IEEE Std 802.1Q-2018 (Revision of IEEE Std 802.1Q-2014)*, pp. 1–1993, 2018.

[5] "IEEE Standard for Local and metropolitan area networks – Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic," *IEEE Std 802.1Qbv-2015 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, and IEEE Std 802.1Q-2014/Cor 1-2015)*, pp. 1–57, 2016.

[6] B. Houtan, M. Ashjaei, M. Daneshtalab, M. Sjödin, and S. Mubeen, "Synthesising Schedules to Improve QoS of Best-Effort Traffic in TSN Networks," in *29th International Conference on Real-Time Networks and Systems*, ser. RTNS'2021. New York, NY, USA: Association for Computing Machinery, 2021, p. 68–77. [Online]. Available: https://doi.org/10.1145/3453417.3453423

[7] R. Serna Oliver, S. S. Craciunas, and W. Steiner, "IEEE 802.1Qbv Gate Control List Synthesis Using Array Theory Encoding," in *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2018, pp. 13–24.

[8] M. Vlk, Z. Hanzálek, K. Brejchová, S. Tang, S. Bhattacharjee, and S. Fu, "Enhancing Schedulability and Throughput of Time-Triggered Traffic in IEEE 802.1Qbv Time-Sensitive Networks," *IEEE Transactions on Communications*, vol. 68, no. 11, pp. 7023–7038, 2020.

[9] A. A. Syed, S. Ayaz, T. Leinmüller, and M. Chandra, "MIP-based Joint Scheduling and Routing with Load Balancing for TSN based In-vehicle Networks," in *2020 IEEE Vehicular Networking Conference (VNC)*, 2020, pp. 1–7.

[10] "What is Factory Automation and what are the benefits for today's IoT enterprises?" https://www.cassianetworks.com/blog/factoryautomation/, [Online; accessed 15-May-2022].

[11] "Time Sensitive Networks for Flexible Manufacturing Testbed - Description of Converged Traffic Types," https://hub.iiconsortium.org/tsn-converged-traffic-types, [Online; accessed 29-March-2022].

[12] 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 the 24th International Conference on Real-Time Networks and Systems*, ser. RTNS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 183–192. [Online]. Available: https://doi.org/10.1145/2997465.2997470