TSN Qbv and Schedule Generation Approaches

Vishavjeet Ghotra, Max Helm*, Benedikt Jaeger*
*Chair of Network Architectures and Services, Department of Informatics
Technical University of Munich, Germany
Email: vishav.ghotra@tum.de, helm@net.in.tum.de, jaeger@net.in.tum.de

***Abstract*—Ethernet is one of the most widely used LAN technologies. This vast application of Ethernet in different fields is due to its features like low cost, flexibility, reliability and easy maintenance. Some application fields like industrial automation require deterministic networks but standard Ethernet is not designed to meet real-time requirements. Time Sensitive Networking (TSN) is a set of standards designed to provide determinism and enable real-time capabilities in Ethernet-based networks. IEEE 802.1Qbv is one of these standards that uses Time-Aware Shaper (TAS) to create Time Division Multiple Access (TDMA) schemes for scheduling and transmitting data in a deterministic manner. This paper provides an overview of IEEE 802.1Qbv and TAS, as well as explains the purpose of schedule generation and compares three different approaches to generate schedules.**

***Index Terms*—time-sensitive networking, time-aware shaper**

1. Introduction

Ethernet is a popular network technology known for its low maintenance requirements and high data transfer rates. However, it does not provide the low latency needed for certain fields such as automotive and network automation that require real-time and deterministic communication [1].

To solve this problem, *Time Sensitive Networking* (TSN) standards were introduced in 2012 to provide standardized real-time mechanisms across Ethernet networks. Some of these standards are IEEE 802.1AS (provides clock synchronization), IEEE 802.1Qbu (frame preemption), IEEE 802.1Qbv (enhancements for scheduled traffic), and IEEE 802.1Qca (Path Control and Reservation). In IEEE 802.1Qbv, incoming frames are assigned to different queues on basis of their traffic class. A gate mechanism called *Time Aware Shaper* (TAS) defines a timed gate for each queue that opens or closes in accordance with the schedule implemented in form of *Gate Control List* (GCL). The GCL contains the configuration of all timed gates (open or closed) and determines which gate should be opened for transmission in a given time slot.

Because IEEE 802.1Qbv does not define any algorithm for schedule synthesis, several algorithms are proposed by researchers that focus on different network configurations and traffic classes. This synthesis problem can be reduced to NP-hard problems like bin-packing [2] and thus, itself is also an NP-hard problem.

The goal of this work is to explain the working of IEEE 802.1Qbv (TAS) and explain the working of three approaches for synthesizing schedules. The remainder of this paper is structured as follows. In Section 2, we introduce relevant research on schedule generation. Section 3 provides information about the working of IEEE 802.1Qbv and categorizes different types of methods for creating schedules. Afterward, we describe three schedule approaches in Section 4. Then we evaluate and compare these approaches in Section 5. Lastly, we conclude in Section 6.

2. Related work

As we delve into the workings of 802.1Qbv TAS, it is important to consider the contributions of prior research on schedule synthesis. In the following section, we review relevant literature in this field.

Steiner [3] introduced an Satisfiability Modulo Theory (SMT) based approach to schedule Time-Triggered (TT) traffic in a network. He formulated the scheduling problem as a set of logical constraints that could be solved by SMT solvers. Hellmanns et al. [4] proposed a hierarchical approach that uses a Tabu Search algorithm to schedule large factory networks. Dürr et al. [5] developed a no-wait scheduling algorithm based on Integer Linear Programming (ILP) and Tabu Search. Berisa et al. [6] presented a heuristic method for improving the schedulability of Audio Video Bridging (AVB) streams

In this work, we first survey the SMT based approach proposed by Craciunas et al. [7] to schedule Scheduled Traffic (ST) frames. Next, we explore the method introduced by Houtan et al. [8] for generating schedules that enhances the Quality of Service (QoS) of best-effort traffic in TSN networks. Lastly, we review the window-based heuristic approach proposed by Reusch et al. [9] to schedule large networks.

3. Background

In this Section, we explain the gate mechanism introduced in 802.1Qbv and the concept of the GCL. Afterward, we introduce two types of approaches for generating schedules.

3.1. Time-Aware Shaper

To enable scheduling of Ethernet frames using IEEE 802.1Qbv standards in time-sensitive networks, network components such as switches that are compatible with Qbv are used. Traffic in a network is classified mainly into

doi: 10.2313/NET-2023-06-1_06

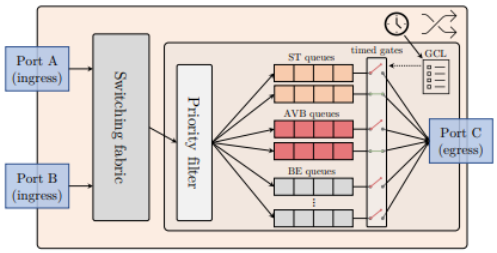

Figure 1: Switch with IEEE 802.1Qbv [6]

three classes: ST, AVB, and Best Effort (BE). ST streams have the highest priority and demand high determinism, whereas BE streams have the lowest priority.

Figure 1 presents a TSN-compatible switch with 2 ingress ports A and B and one egress port C. The function of the switching fabric is to perform mapping of these ingress ports to egress ports for a given stream. There are 8 queues assigned to 3 traffic classes. ST traffic is assigned to higher-priority queues and lower-priority queues are dedicated to BE traffic. The priority filter decides in which queue a frame of the given stream will be enqueued. TAS introduces a timed gate in each queue. Each timed gate has two states: open (1) or closed (0). Frames are dequeued from a queue in a FIFO manner. If gates of multiple queues are opened simultaneously, the frames from the highest-priority queue are transmitted. The state of each timed gate and the time of opening/change in the state of these gates is encoded in a GCL [6]. Each entry in the GCL consists of 2 elements, where the first element is the time relative to the start of the GCL and the second is the state of the gates represented as a bitmask. For example, the entry "T_1: 10000000" means that only the gate of the first queue is opened at the relative time T_1. GCL is a cyclic schedule, i. e., the entries in GCL repeat themselves after a predefined time period T_{cycle}. Furthermore, the IEEE 802.1AS standard is used to synchronize the clocks of all TSN switches in the network [5].

TAS defines the whole gate mechanism that allows or denies the transmission of frames by opening or closing the corresponding queues on the basis of the GCL, but it does not define how the entries in the GCL are created, i. e., it does not define an algorithm that decides the state of the gates at a given time. Therefore, several researchers proposed algorithms that allow optimal communication for different parameters, e. g. some algorithms aim to achieve minimum latency for ST streams without taking the latency of lower-priority streams into consideration [7] while some tend to minimize the maximum end-to-end delay of BE streams [8].

3.2. Schedule Creation

The approaches for creating schedules can be grouped into two categories: exact approaches and heuristic approaches. Exact approaches use methods like SMT, Optimization Modulo Theorem (OMT), and ILP. In such approaches, a constrained satisfaction problem is constructed from the scheduling problem and then solved by methods like ILP or SMT. One of the main advantages of these approaches is that the results generated from these approaches are provably optimal. But the scheduling problems are NP-hard, thus calculating exact solutions for large networks demands high computation time.

Heuristic approaches solve this problem and speed up the calculations by finding sub-optimal solutions. But the results generated by these approaches are not provably optimal. Many of these approaches employ heuristic techniques such as Tabu Search and Simulated Annealing.

4. Approaches for schedule creation

In this Section, we explore three approaches for schedule creation. First, we explore the SMT-based approach to schedule ST traffic by Craciunas et al. [7]. Afterward, we analyze another SMT-based approach to improve the QoS of BE traffic proposed by Houtan et al. [8]. Finally, we inspect the window-based approach devised by Reusch et al. [9] to schedule traffic in large networks.

4.1. SMT Based ST Scheduling

Craciunas et al. [7] formulated the scheduling problem as a set of constraints. These constraints are then passed to the SMT solvers to find the optimal values for variables like frame offset such that all constraints are satisfied.

Many factors like sharing of the same queue by frames of different streams affect the deterministic behavior of a network. To avoid this, the following constraints were defined in [7].

Frame Constraint. This constraint assures that offset of each frame is greater than or equal to 0 and less than or equal to the frame period and thus, guarantees that the frame transmission will be completed before the start of the next period. [10].

Link Constraint. Only one frame can be transmitted at a time on a given physical link. This constraint enforces that no two frames directed to the same physical link overlap each other [10].

Flow Transmission Constraint. This constraint enforces that frames of a stream follow the routed path of the stream in a specific order. In other words, it assures that a frame can only be sent on the next link in the path after it has been fully received on the previous link [7].

End-to-End Constraint. The time between a stream being transmitted by the sender and received at the destination must be less than or equal to the specified duration to avoid any deadline misses for ST streams [7].

If frames of two different streams arrive at the same time, the order in which the frames are placed in queues is not clear. As shown in Figure 2(a), the frames from both flows may get enqueued in any order and therefore, they may be interleaved in any combination on the egress port. Conditions must be defined to avoid this interleaving by either placing such frames in different queues (Figure 2(b)) or maintaining the intended order and transmission time for all frames if the streams are placed in the same queue (Figure 2(c)) [7].

Stream Isolation Constraint. Two frames `F1` and `F2` of different flows are planned to arrive and be assigned to a queue in a particular order, e. g., `F1` before `F2`. If frame `F1` is lost and is not attached to the queue, frame `F2` will get transmitted in the time slot reserved for `F1`, which

doi: 10.2313/NET-2023-06-1_06

(a) Interleaving streams in egress queue. (b) Isolation of streams in different queues. (c) Isolation of stream arrival.

Figure 2: Flow interleaving and isolation within an egress port [7]

results in non-determinism [7]. To avoid this problem, this constraint enforces that if a frame of a given flow is queued, no frame of other flows is allowed to enter the queue until all frames from the given flow are fully transmitted to the output port [7].

Frame Isolation Constraint. The constraint specified above decreases the search space for the schedules. This new constraint loosens it and enforces that the frames from the second flow are now only required to wait for the frames present in the queue and not for *all* frames of the first flow to be dispatched to the output port [7].

The concept of using Satisfiability Modulo Theories (SMT) solvers to create schedules was first introduced by Steiner [3]. SMT solvers provide a model for the given context when a set of variables and constraints in the context is satisfiable. This model represents one of the potentially many solutions to the given constraints. The goal of this 802.1Qbv scheduling algorithm is to find the optimal values for the frame offsets and queue assignments for each egress port of routed flows in the network such that all constraints explained above are satisfied. This approach involves scheduling each flow one at a time by adding the flow's variables and constraints to the SMT context and attempting to solve the problem [7].

If a solution is found, the variables for the flow are replaced by the constant value provided by the SMT model. This repeats until all flows are scheduled (a solution found) or the solver determines that it is not possible to satisfy newly added constraints in the context. If an unfeasible step is encountered, the SMT context is backtracked, and the last added flow is removed. The next optimal values of variables for the removed flow are then determined before returning to the unfeasible step. This backtracking algorithm repeats until a complete solution for all flows is found or it is found that solution does not exist after checking every possible combinations of values using backtracking. In the latter case, a suboptimal solution is returned that is able to schedule the maximum number of flows [7].

4.2. QoS Improvement

The approach explained above addresses the scheduling of ST streams but does not focus on BE traffic class. The approach in [8] is an enhancement that uses all constraints defined in [7] and proposes a new set of constraints and objective functions to create schedules for ST traffic while improving QoS for BE traffic.

The concept of slack is introduced to accommodate BE frames between consecutive ST frames. Slack is a period of time after the transmission of an ST frame during which no other ST frames can use the bandwidth of the link [8].

Porous Link Constraint. This is a modified form of the link constraint explained above. The link constraint avoids the timely overlap of frames on the same link. This constraint has been modified to take slack into account and avoids overlapping of frames along with their slacks [8].

Slack Size Constraint. This constraint verifies the size of slack used for each frame scheduled on the link. The used slack size must be greater than or equal to zero, but it must also be less than or equal to the difference between the frame period and its transmission time [8].

Hop Slacks Constraint. This constraint bounds the total amount of slacks allowed on the link [8].

Equal Link Constraint. This is an optional constraint that enforces equal slack sizes for all frames on a link [8].

Three objective functions were introduced in addition to these constraints.

Maximization. This optimization objective function attempts to schedule the transmission of ST frames close to their deadlines, while still packing them together. This maximizes the allocation of available bandwidth for the transmission of BE frames at the start of the schedule and avoids deadline misses for these frames [8].

Sparse Schedule. This function seeks to maximize the overall slack between subsequent ST frames that are scheduled to be transmitted on the same link. In other words, it helps to create gaps between subsequent ST frames to fit BE frames to minimize the delay [8].

Evenly Sparse Schedule. This function modifies the sparse schedule objective function by asserting the equal porous constraint specified above to create equally-sized slacks on the links [8].

An SMT solver checks the satisfiability of these constraints and returns one solution if they are satisfiable.

4.3. Window-Based Heuristic Approach

Real-time communication is critical in large industrial networks. SMT-based approaches are not ideal for scheduling these large networks, because the large constrained problems formulated have exponential runtime w. r. t. the number of flows. Thus, heuristic approaches are widely used for scheduling large networks, as they provide solutions in a shorter period of time that may or may not be optimal.

Reusch et al. proposed a heuristic approach in [9] to create schedules. The main goal of this work is to find the optimal window length and period so that no deadline for an ST flow is missed and the total available bandwidth of all ports in the network is maximized. The purpose of maximizing available bandwidth is to allow room for lower-priority traffic.

A window of a queue refers to the time interval for which the timed gate of the queue is opened and allowed to transmit frames. A cost function is defined for each port that determines the number of windows active in

a given period. These cost functions are added together to get the overall cost function. A heuristic algorithm is developed to find an initial solution that minimizes this overall cost function. A solution is considered *valid* if all flows have a finite worst-case delay. If the initial solution provided by the heuristic method is *valid*, it might be cost-optimal but it does not imply that all deadlines are met. Thus, a window optimization algorithm checks if the initial solution satisfies the deadlines of all flows. If these deadlines are met, then the initial solution is returned as the final solution.

Otherwise, the infeasible flows are sorted in descending order based on the percentage by which they exceed their deadlines. The flow with the largest percentage is optimized first. Optimizing a flow involves adjusting the period of all ports on its route through an iterative process. This is done by increasing or decreasing the period of every window in each port by the same amount in each iteration. If the flow is not feasible after adjusting the periods, they are decreased for all ports. If the flow is feasible, the periods are increased until the solution becomes infeasible or the periods no longer change. The motive here is to find the maximum period for each port that allows all flows to meet their deadline [9].

Ultimately, we either get a feasible solution or there could be a solution that exists but cannot be discovered through this method.

5. Comparison

In this section, we discuss the results achieved by the three approaches. We present and compare the results obtained using these approaches.

5.1. ST Scheduling

Z3 SMT/OMT solver and Yices v2.4.2 (64bit) were used for evaluation. The experiments were conducted on a 64bit 4-core 3.40GHz Intel Core-i7 PC with 4GB memory and a 5-hour time-out value. Synthetic configurations were used to analyze the scalability and schedulability of the networks, based on three predetermined network topologies. These topologies ranged from 3 end-systems connected to one switch to 7 end-systems connected through 5 switches. To achieve higher utilization on the links, the size of the topologies was kept relatively small compared to the number of flows. The results indicated that the runtime increased exponentially with an increase in the number of flows and frames scheduled, while the period set has a significant impact on scalability. The flow isolation approach consistently outperformed the frame isolation approach, with an average 13% faster runtime at the expense of schedulability. Further experiments confirmed this trend, with the flow isolation approach being faster on average than the frame isolation approach [7].

5.2. QoS Improvement

For the purpose of evaluation, a multi-hop network with six end stations was used. Different scenarios were selected that featured different ratios of ST and BE streams. In addition to the three objective functions explained in the Section 4.2, the minimization objective function is used to schedule ST streams. The minimization objective function minimizes the total offset of ST streams. The end-to-end delay of BE streams, deadline misses, and runtime were measured for each objective function. Moreover, Z3 SMT/OMT solver was used to solve the constrained problem and find a feasible solution.

The minimization objective function resulted in the longest end-to-end delay for BE streams in all scenarios, while the maximization objective function gave the best results. The reason is that the maximization function schedules ST frames as close to their deadlines as possible, which creates room for BE frames to be scheduled. The sparse and evenly sparse objective functions also produced lower end-to-end delays due to the slack, that is built in to accommodate BE frames.

The minimization and maximization functions had a large number of missed deadlines, while the sparse schedule function and evenly sparse schedule functions had no missed deadlines in any of the scenarios. Both the sparse schedule function and evenly sparse schedule function also had the best runtime. In scenarios with fewer ST streams, the maximization function was faster than the minimization function. However, as the number of ST streams increased, the maximization function became slower and was eventually outperformed by the minimization function [8].

5.3. Window-Based Heuristic Approach

Seven test cases were designed based on industrial application requirements to evaluate the effectiveness of the approach. Moreover, the greedy randomized adaptive search procedure (GRASP) metaheuristic from [11] and Strict Priority (SP) policy were used for comparison. Four aspects were considered in the comparison: worst-case delay, mean cost, calculation time, and the number of infeasible flows.

The results showed that although the proposed algorithm had a better execution time than GRASP, it was outperformed by GRASP in the other three aspects. The SP algorithm had the best runtime, but it caused a significant delay for other traffic classes. GRASP was able to schedule all flows in all test cases, but required exponential time to get solutions in some cases. Moreover, GRASP was less robust and more difficult to adapt to changing conditions in real-time applications, compared to the proposed algorithm [9].

5.4. Comparison

As shown in the Table 1, ST Scheduling [7] solely focuses on scheduling ST frames, while the other two approaches take both ST and BE frames into consideration. Additionally, ST Scheduling [7] has the highest runtime among the three approaches. For small networks, QoS Improvement [12] exhibits the most efficient runtime, whereas Window-based Heuristic [9] has the best runtime for larger networks but as a tradeoff, it returns suboptimal solution, unlike the other two approaches that always return optimal solution.

Therefore, we can conclude that out of the three approaches, the third approach is optimal for large networks,

Name	Solution Approach	ST	BE	Nodes	Streams	Runtime (s)	Solution Optimal?
ST Scheduling [7]	SMT	✓	×	12	1000	<18000	✓
QoS Improvement [12]	SMT	✓	✓	8	10	0.37-1153.52	✓
Window-based Heuristic [9]	Heuristic	✓	✓	402	316	∼10.52	×

TABLE 1: Overview of the approaches discussed in this paper [13]

whereas for small networks, the second approach is the most suitable approach.

6. Conclusion and Future Work

In this paper, we described the working of the gate mechanism in IEEE 802.1Qbv and explored and analyzed three different approaches for creating schedules. The first approach scheduled ST frames without taking other traffic classes in consideration. The second approach aimed to optimize the scheduling of BE frames while meeting the deadlines of ST frames for small networks using SMT solvers. The last approach used a heuristic algorithm to schedule ST frames and maximize the available bandwidth for BE frames in large networks. All approaches succeeded in their objectives. We also compared these approaches and concluded that the third approach had the best runtime at the expense of the optimality of the solution returned, whereas the second approach outperformed the first approach for small networks.

Future work on the first and second approaches could focus on reducing the complexity of the scheduling problem by trying to remove or loosen some constraints while still meeting the deadlines of all flows. One possible extension to the window-based approach could be further optimization using new heuristic techniques to minimize the overall delay.

References

[1] "Is Ethernet Deterministic?" https://polytron.com/blog/is-ethernet-deterministic-does-it-matter-part-1/, [Online; accessed 19-December-2022].

[2] 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)*. IEEE, 2020, pp. 1–7.

[3] W. Steiner, "An Evaluation of SMT-based Schedule Synthesis for Time-Triggered Multi-Hop Networks," in *2010 31st IEEE Real-Time Systems Symposium*. IEEE, 2010, pp. 375–384.

[4] 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)*. IEEE, 2020, pp. 1–8.

[5] F. Dürr and N. G. Nayak, "No-Wait Packet Scheduling for IEEE Time-Sensitive Networks (TSN)," in *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, 2016, pp. 203–212.

[6] A. Berisa, L. Zhao, S. S. Craciunas, M. Ashjaei, S. Mubeen, M. Daneshtalab, and M. Sjödin, "AVB-aware Routing and Scheduling for Critical Traffic in Time-sensitive Networks with Preemption," in *Proceedings of the 30th International Conference on Real-Time Networks and Systems*, 2022, pp. 207–218.

[7] S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner, "Scheduling Real-Time Communication in IEEE 802.1 Qbv Time Sensitive Networks," in *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, 2016, pp. 183–192.

[8] 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*, 2021, pp. 68–77.

[9] Reusch, Niklas and Zhao, Luxi and Craciunas, Silviu S. and Pop, Paul, "Window-based schedule synthesis for industrial ieee 802.1qbv tsn networks," in *2020 16th IEEE International Conference on Factory Communication Systems (WFCS)*, 2020, pp. 1–4.

[10] S. S. Craciunas and R. S. Oliver, "Combined Task-and Network-Level Scheduling for Distributed Time-Triggered Systems," *Real-Time Systems*, vol. 52, no. 2, pp. 161–200, 2016.

[11] M. L. Raagaard and P. Pop, "Optimization Algorithms for the Scheduling of IEEE 802.1 Time-Sensitive Networking (TSN). Tech. Univ. Denmark, Lyngby," Denmark, Tech. Rep, Tech. Rep., 2017.

[12] S. Martello and P. Toth, "Bin-packing Problem," *Knapsack problems: Algorithms and computer implementations*, pp. 221–245, 1990.

[13] T. Stüber, L. Osswald, S. Lindner, and M. Menth, "A Survey Of Scheduling In Time-Sensitive Networking (TSN)," *arXiv preprint arXiv:2211.10954*, 2022.