Synthesising Schedules to Improve QoS of Best-effort Traffic in TSN Networks

Bahar Houtan, Mohammad Ashjaei, Masoud Daneshtalab, Mikael Sjödin, Saad Mubeen
{bahar.houtan,mohammad.ashjaei,masoud.daneshtalab,mikael.sjodin,saad.mubeen}@mdh.se
Mälardalen University, Sweden

ABSTRACT

The IEEE Time-Sensitive Networking (TSN) standards' amendment 802.1Qbv provides real-time guarantees for Scheduled Traffic (ST) streams by the Time Aware Shaper (TAS) mechanism. In this paper, we develop offline schedule optimization objective functions to configure the TAS for ST streams, which can be effective to achieve a high Quality of Service (QoS) of lower priority Best-Effort (BE) traffic. This becomes useful if real-time streams from legacy protocols are configured to be carried by the BE class or if the BE class is used for value-added (but non-critical) services. We present three alternative objective functions, namely Maximization, Sparse and Evenly Sparse, followed by a set of constraints on ST streams. Based on simulated stream traces in OMNeT++/INET TSN NeSTiNg simulator, we compare our proposed schemes with a most commonly applied objective function in terms of overall maximum end-to-end delay and deadline misses of BE streams. The results confirm that changing the schedule synthesis objective to our proposed schemes ensures timely delivery and lower end-to-end delays in BE streams.

CCS CONCEPTS

• **Computer systems organization → Real-time systems**; **Embedded systems**.

KEYWORDS

Time sensitive networking, TSN, schedule optimization, scheduled traffic.

ACM Reference Format:
Bahar Houtan, Mohammad Ashjaei, Masoud Daneshtalab, Mikael Sjödin, Saad Mubeen. 2021. Synthesising Schedules to Improve QoS of Best-effort Traffic in TSN Networks. In *29th International Conference on Real-Time Networks and Systems (RTNS'2021), April 7–9, 2021, Nantes, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3453417.3453423

1 INTRODUCTION

Recent functionality advancements and innovation in the applications of embedded systems, especially in the automotive and automation domains, require high-bandwidth and low-latency communications. To meet these communication requirements, the IEEE 802.1 TSN task group[1] took an initiative to develop a set of Time-Sensitive Networking (TSN) standards over the switched Ethernet. Notable features of TSN include: clock synchronization (IEEE Std 802.1AS-2020) [17], Time-Aware Shaper (TAS) for offline scheduled time-triggered traffic (IEEE 802.1Qbv), Credit-based Shaper (CBS) and bandwidth reservation (IEEE Std 802.1Qav), frame preemption (IEEE Std 802.1Qbu), among others, which are rolled into the IEEE 802.1Q-2018 standard [16].

The queuing and forwarding mechanism in the TSN standards distinguishes between critical and non-critical traffic classes. The critical traffic classes are defined as Classes A and B. Where Class A has a higher priority than Class B. Whereas, the non-critical traffic class is defined as the best-effort (BE) traffic class with the lowest priority. In addition, the scheduled traffic (ST) enhancement in the TSN standard defines an ST traffic class with strict temporal isolation. The temporal isolation is achieved by a gate mechanism following the TAS mechanism, where the traffic transmission is allowed only when the gate for a queue is open. The gates operation follows a pre-defined gate control list (GCL) that repeats periodically defining which gate on the queues should be open at each time slot.

Currently the use of TSN in industrial systems is gaining a significant momentum [19, 20]. However, redesigning and replacing the existing communication systems is a relatively costly process. For example, many industrial systems use different Ethernet communication protocols, e.g., EtherCAT[2], with very different interfaces and message format compared to that of TSN. Therefore, one of the main challenges is to replace the existing networks with TSN making as little as possible changes in the end stations. Within this context, a non-trivial task is to map the existing network traffic to the TSN traffic classes such that the previous properties, e.g., jitter and delay, still hold. One of the straightforward solutions is to map all time-critical traffic into the ST traffic class to ensure the low jitter and low-latency transmission for the traffic. However, there are traffic that do not have strict deadlines to meet, yet achieving an acceptable level of Quality of Service (QoS) for them is expected. For example, diagnostic signals, network status checks and software update signals are among less time-critical traffic, which usually have soft real-time requirements. Frequent violation of the timing requirements for such traffic can hamper the corresponding functionalities. These type of traffic are often mapped to the BE class that requires no change in the network interfaces of the end stations.

Many works in the literature address the problem of ST traffic scheduling, which is known as an NP-hard problem. Most of the solutions formulate the problem as an optimization problem (essentially a bin-packing problem) that is solved by defining a set

RTNS'2021, April 7–9, 2021, Nantes, France

ACM ISBN 978-1-4503-9001-9/21/04…$15.00
https://doi.org/10.1145/3453417.3453423

[1] https://1.ieee802.org/tsn/
[2] https://www.ethercat.org/

of constraints, e.g., [4, 26]. The main requirement for the solver in these solutions is to provide a feasible schedule taking jitter and delay of the ST traffic into account. There are also a few works that consider different optimization goals such as maximizing porosity on the network links [25] to allow fast schedule updates in the case of faults in the network. There are a few solutions based on meta-heuristic algorithms, e.g., the solution in [22] that uses the genetic algorithm. Nevertheless, the proposed ST scheduling solutions in the literature strictly consider the timing requirements for the ST traffic, with an exception in [15] that considers schedulability of classes A and B while defining a routing mechanism for the ST traffic. To the best of our knowledge, none of the existing ST scheduling solutions take into account QoS for the BE traffic that has soft real-time requirements, e.g., legacy diagnostic signals or network status checks. In this paper, we propose a solution to synthesize feasible schedules for the ST traffic, while providing a high level of QoS for the BE traffic. To the extent of our knowledge, existing analytical schedulability analysis methods do not provide support for schedulability analysis of low priority BE traffic in IEEE TSN standards. Therefore, we use a simulation method to evaluate our proposed solutions.

The main contributions in this paper are as follows:

- we mathematically model and present optimization constraints to consider the QoS for the BE traffic;
- we propose new optimization objective functions to obtain a feasible ST schedule while improving the QoS of the BE traffic;
- we show that the commonly used objective function (minimizing the ST offsets) in the existing works leads to poor QoS for any lower priority traffic class; and
- we evaluate the proposed solutions with a set of experiments using the OMNeT++ simulation platform. We use Z3 SMT/OMT solver to implement the proposed constraints and objective functions. Then, we compare the proposed solutions with the existing solutions to show their effectiveness in providing a high level of QoS for the BE traffic.

2 BACKGROUND AND RELATED WORK

2.1 The gate mechanism in TSN

The IEEE 802.1Qbv standard (rolled into IEEE 802.1Q-2018) defines a gate mechanism following the TAS mechanism to allow the ST traffic being transmitted without any interference. According to this mechanism, the class types are recognized by a 3-bit Priority Code Point (PCP) value in the IEEE TSN 802.1Q compatible frame headers. The example in Figure 1 illustrates a simplified gate mechanism, in which Q_0, Q_1, Q_2 and Q_3 correspond to classes ST, A, B and BE, respectively. A Gate Control List (GCL) contains an array of time-stamped vectors to control the output of each queue. The list is repeating periodically. At the specified times in the time-stamped vector, the data in the vector is sent to the gate drivers, which enable or disable the transmission of the associated class. In this example, the value "1" represents open state to enable transmission, while "0" closes the gate to disable the transmission in certain queues. Note that only classes A and B undergo the CBS shaper.

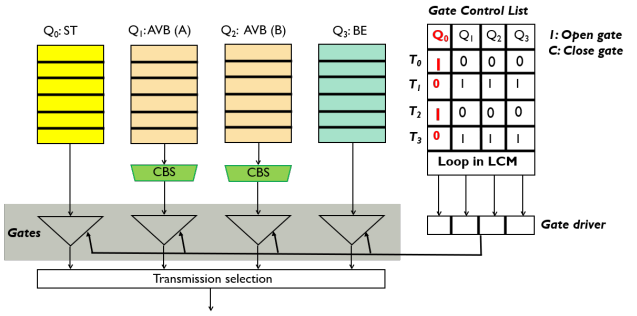


Figure 1: Gate mechanism block diagram.

2.2 Schedule Synthesis

The schedule synthesis approaches can be grouped into two branches as shown in Figure 2: (i) Satisfiability Modulo Theorems (SMT) and Optimization Modulo Theorem (OMT), and (ii) heuristics and meta-heuristic approaches. Most of the works in schedule synthesis were using SMT solvers to search all the possibilities for optimal schedule. SMT solvers are automated theorem provers to prove satisfiability and validity of first-order logical statements by examining each possible combination of variables in the search space. The output of the SMT solvers is a random value for the SMT variable that satisfies the specified constraints. To improve scalability, some works have either used a new type of optimized SMT solver called OMT or Mixed-Integer Programming (MIP). The OMT Solvers have become popular as they enable optimizing SMT Solver variables based on user-defined objective functions and allow to solve linear optimization problems over satisfiable SMT formulas [1].

The majority of the previous works were concentrated on the synthesis of feasible schedules for the high priority critical traffic based on a work by Steiner et al. [29]. In the series of works by Craciunas et al. [2–6, 27] the network was modelled by directed graphs that indicate relations between end stations and switches via links. In [4], schedule optimization constraints were derived from TTEthernet scheduling constraints to address the specifications of IEEE 802.1Qbv. The generic constraints of the solution include: frame, link, stream and end-to-end delay timing constraints. Schneider et al. [7, 28, 31] introduced an SMT-based ST schedule synthesizer tool, named TSNSched, which modeled TSN network via data structures.

Steiner et al.[30] introduced five strategies to enable porous (or blank) schedule synthesis. These strategies include: a) a priori schedule variation to force blank schedules, b) a posteriori schedule variation with less computational time-complexity, c) combined schedule variation in order to assure the least overhead, d) interpretation approach to change the timeline into equal time slots defining minimum size for blanks, and e) combined schedule variation with the above steps [30].

In the same line of research, the work by Pozo et al. [25] investigated the generation of porous link schedules, which can help in achieving extra time spans to react to possible link failures. The proposed method takes advantage of a link reparation post-processing procedure to find alternative paths that do not pass through the failed links. In addition, the work by Gavrilut et al. [15] proposed a two-stage approach for scheduling ST traffic considering an

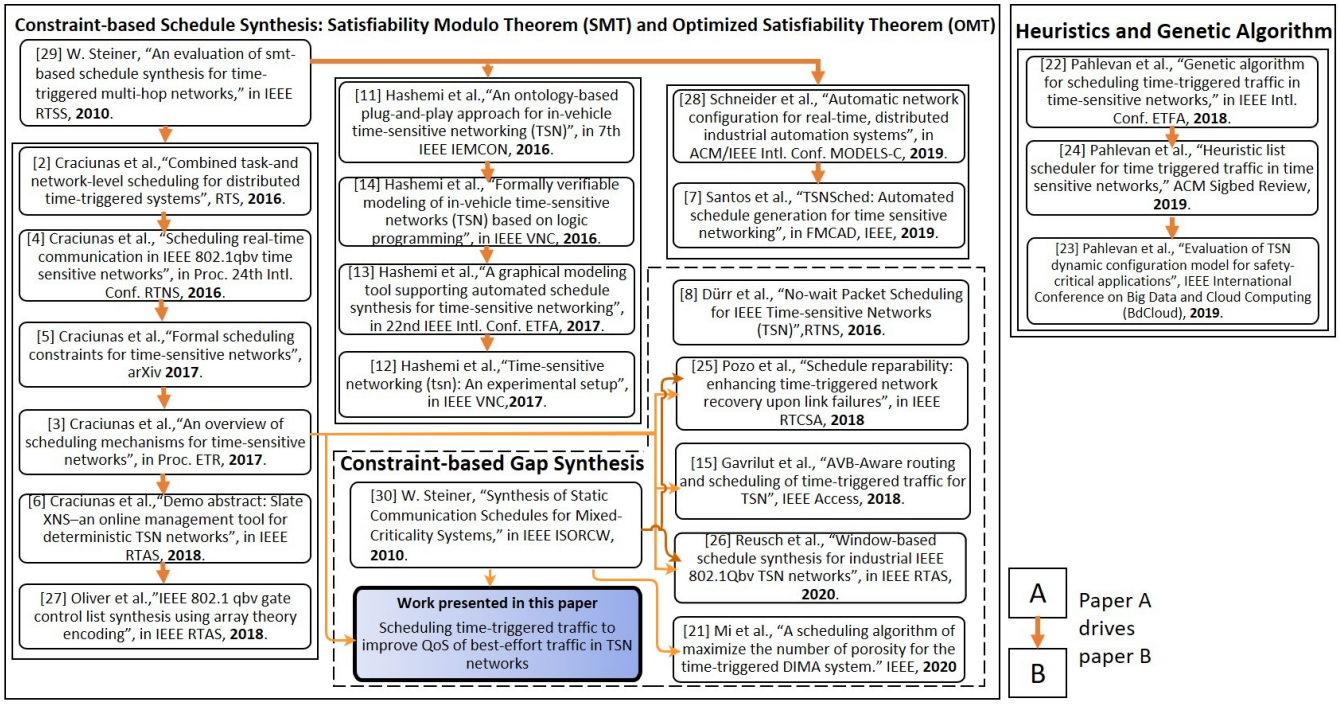


Figure 2: Overview of the schedule synthesis approaches.

optimum routing for the ST traffic, while taking into account the schedulability of the AVB traffic (i.e., A and B).

A series of works considered different approaches than only fitting the ST traffic into time-slots. For example, the work by Reusch et al. [26] proposed a window-based schedule synthesis approach that maximizes the bandwidth allocated for lower priorities. Their solutions require exclusion of deadline miss constraints for the ST schedule. Therefore, a post-processing iterative optimization heuristic was presented to shift slack windows in a manner to save deadline misses due to schedules made by a gap synthesis. Thereafter, the proposed method is evaluated by the worst-case delay analysis proposed in [33]. Another similar work by Mi et al. [21] considered two-stage gap synthesis. The purpose of the work is to create porous schedules to reduce the transmission delay and jitter of other real-time traffic. Moreover, the works by Hashemi et al. [10–14] adopted an ontology-based approach, utilizing logic programming in Prolog programming language, to show properties and the relations between network entities and components.

Considering the solutions based on heuristics and meta-heuristic we can find several works in the research community. For instance, the work by Pahlevan et al. [22–24] investigated heuristic/GA-based solutions to overcome time consuming constraint solving processes. However, the proposed solution was not compared with the SMT-based solutions. Moreover, Dürr et al. [8] proposed a Tabu search-based approach to find feasible schedules of the ST traffic. The authors proposed a bandwidth utilization by means of scheduling the ST frames close to each other.

Although there are several solutions to schedule ST traffic in TSN networks, as reviewed above, most of them considered only the timing properties of the ST traffic in their scheduling solutions. The exception is the work in [15] and [25], where in the former the schedulability of AVB traffic was considered and in the latter repairability of the links after failures was the main aim. In our work, we propose a set of constraints and three objective functions to schedule the ST traffic while improving the QoS of the BE traffic in the network considering that they have soft real-time properties due to legacy network support. We base our solution on the work presented in [4].

3 SYSTEM MODEL

The system model considered in this paper is inspired by the system model in [4]. Note that the paper focuses only on the ST and BE traffic classes. The network model, as shown in Figure 3, is represented by a directed graph, $G(V, L)$. The vertices V represent end stations and switches in the network. The links, represented by L, are modeled as two-directional edges connecting a set of vertices. A two-directional link between the end stations $v_a \in V$ and $v_b \in V$ is represented by $[v_a, v_b] \in L$. The link attributes are denoted by the tuple $\langle speed, d, mt \rangle$, where the parameters $speed$, d and mt indicate the link's speed, link propagation delay and macrotick, respectively. The model is flexible such that each link can accept different speed and propagation delay. The macrotick serves for the scalar granularity of the schedule time-line. For example, macrotick of $1\mu s$ indicates that the schedule time unit is in the order of microseconds.

The set of streams in the network are represented by S. A unicast stream with ID i belonging to S is represented by s_i, i.e., $s_i \in S$. The path taken by a stream from the source end station, v_s, to the destination end station, v_d, through a set of switches is

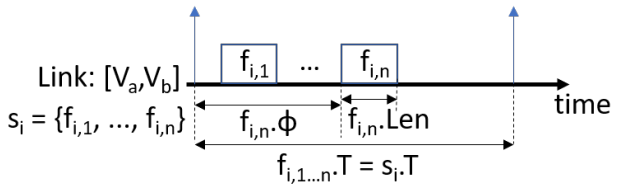


Figure 3: System model parameters.

shown by $s_i.path = [[v_s, v_{s+1}], \ldots, [v_{d-1}, v_d]]$, where $[v_s, v_{s+1}]$ and $[v_{d-1}, v_d]$ indicate the first and last links along the path of the stream. The stream's attributes are defined by the tuple, $\langle e2e, Size, T, p \rangle$. Where, $e2e$ is the timing constraint on the end-to-end delay of the stream, i.e., the maximum end-to-end delay shall not exceed the value assigned to the e2e attribute. $Size$ is transmission time of the stream on the link, which is the time it takes to transmit a full stream on the link in the scale of macroticks. The parameters, T and p, are stream's period and priority, respectively. The parameter, p, obtains a value from the set $p = \{BE, ST\}$, where BE shows the best-effort class and ST shows the ST class. Note that in case of BE class, T is the minimum inter-arrival time and $e2e$ is the deadline for the BE stream.

A stream may contain n number of frames, i.e., $s_i = \{f_{i,1}, \ldots, f_{i,n}\}$. A frame at the hop between the end station or switch v_a and the end station or switch v_b is represented by $f_{i,j}^{[v_a, v_b]}$, where the first subscript i represents the stream ID and the second subscript j represents the frame ID. Because a stream may contain large data, according to the Ethernet protocols, the data has to be fragmented into frames. The maximum allowed Ethernet frame is limited by the Maximum Transmission Unit (MTU). In this model, each stream can be fragmented into several frames. Note that if the stream size does not exceed the MTU size, then all data is fitted in one frame. Each frame has the attributes $\langle T, Len, \phi \rangle$, where T is the frame's period inherited from the parent stream. The transmission time of the frame over the link $[v_a, v_b]$ is represented by $f_{i,j}^{[v_a, v_b]}.Len$. If a stream belongs to the class ST, $f_{i,j}^{[v_a, v_b]}.\phi$ represents the offset of j^{th} frame of the i^{th} stream on the link $[v_a, v_b]$. There are no offsets for BE frames, hence ϕ is not applicable for the BE frames. Moreover, each BE frame is assumed to arrive at the start of its period, which in turn generates the worst-case situation for the BE frames. The parameters for the frames belonging to a ST stream and their attributes are shown in Figure 3 for one period of the stream s_i over one link $[v_a, v_b]$.

4 NETWORK CONSTRAINTS

In this section, we first give an overview of the existing optimization to schedule ST traffic in TSN networks, then we present the proposed additional constraints to improve the QoS of the BE traffic in TSN networks.

4.1 Existing Constraints

We base our work on the ST schedule synthesis constraints in [4] that defines six different constraints to find a feasible schedule. These constraints include: (i) frame constraint, (ii) link constraint, (iii) stream constraint, (iv) end-to-end constraint, (v) stream isolation constraint, and (vi) frame isolation constraint. Below, we present these constraints in detail.

4.1.1 Frame Constraint. This constraint ensures that the offset of each ST frame is at or after time zero. Also, the transmission of the frame must be finished at or before the start of its next period.

$$\forall s_i \in S, s_i.p = ST, \forall [V_a, V_b] \in L, \forall f_{i,j}^{[V_a,V_b]} \in s_i^{[V_a,V_b]} : (f_{i,j}^{[V_a,V_b]}.\phi >= 0) \wedge (f_{i,j}^{[V_a,V_b]}.\phi <= f_{i,j}^{[V_a,V_b]}.T - f_{i,j}^{[V_a,V_b]}.Len) \quad (1)$$

4.1.2 Link Constraint. The constraint on links guarantees that only one frame at a time can be transmitted on a link within the path taken by the stream;

$$\begin{aligned} &\forall s_i, s_j \in S, s_i.p = ST, s_j.p = ST, i \neq j, \forall [V_a, V_b] \in L, \\ &\forall f_{i,k}^{[V_a,V_b]} \in s_i^{[V_a,V_b]}, \forall f_{j,l}^{[V_a,V_b]} \in s_j^{[V_a,V_b]}, \\ &\forall \alpha \in [0, \tfrac{lcm(s_i.T, s_j.T)}{s_i.T} - 1], \forall \beta \in [0, \tfrac{lcm(s_i.T, s_j.T)}{s_j.T} - 1] : \\ &(f_{j,l}^{[V_a,V_b]}.\phi + (\beta \times f_{j,l}^{[V_a,V_b]}.T) + f_{j,l}^{[V_a,V_b]}.Len \leq \\ &f_{i,k}^{[V_a,V_b]}.\phi + (\alpha \times f_{i,k}^{[V_a,V_b]}.T)) \vee \\ &(f_{i,k}^{[V_a,V_b]}.\phi + (\alpha \times f_{i,k}^{[V_a,V_b]}.T) + f_{i,k}^{[V_a,V_b]}.Len \leq \\ &f_{j,l}^{[V_a,V_b]}.\phi + (\beta \times f_{j,l}^{[V_a,V_b]}.T)) \end{aligned} \quad (2)$$

where, α and β are indices to calculate the offsets of multiple frames within the hyper periods of the streams. The hyper period is calculated by the function $lcm()$ that receives the streams pair by pair as inputs and gives the least common multiple of the frames' periods.

4.1.3 Stream Constraint. In order to constrain the orderly transmission and reception of frames within a stream through the path, the sequence of the frames should be considered according to Eq. (3);

$$\begin{aligned} &\forall s_i \in S, s_i.p = ST, \forall [V_a, V_x], [V_x, V_b] \in L, \\ &f_{i,j}^{[V_a,V_x]} \in s_i^{[V_a,V_x]}, f_{i,j}^{[V_x,V_b]} \in s_i^{[V_x,V_b]} : \\ &((f_{i,j}^{[V_x,V_b]}.\phi \times [V_x, V_b].mt) - [V_x, V_b].d - \delta) \geq \\ &((f_{i,j}^{[V_a,V_x]}.\phi + f_{i,j}^{[V_a,V_x]}.Len) \times [V_a, V_x].mt) \end{aligned} \quad (3)$$

The stream constraint is dependent on synchronization of clocks in the source and destination end stations. Where, δ is the synchronization factor, denoting the worst-case drift between the clocks. We assume identical clocks in the source and destination end stations.

4.1.4 End-to-end Constraint. In order to meet the timing requirements of ST streams, the ST streams must not be delivered after their deadlines. Therefore, we need to ensure that the stream's last frame is delivered to the destination end station before the end-to-end deadline. Additionally, the constraint in Eq. (4) takes into account the delay of the links in the path to the destination end station and the worst-case difference between macro ticks of the local clocks of the source and destination end stations. Note that $f_{i,1}$ and $f_{i,|s_i|}$ denote the first and last frames in the stream, s_i. Moreover, $[v_s, v_{s+1}]$ defines the first link in the stream from the source while $[v_{d-1}, v_d]$ represents the last link in the stream to

the destination end stations. Hence, $f_{i,|s_i|}^{[v_{d-1},v_d]}$ represents the last frame in the stream s_i at the destination end station.

$$\begin{aligned}&\forall s_i \in S, s_i.p = ST, f_{i,j} \in s_i : \\ &(f_{i,1}^{[v_s,v_{s+1}]}.\phi \times [v_s,v_{s+1}].mt) + s_i.e2e \geq \\ &(f_{i,|s_i|}^{[v_{d-1},v_d]}.\phi + f_{i,|s_i|}^{[v_{d-1},v_d]}.Len) \times [v_{d-1},v_d].mt\end{aligned} \tag{4}$$

4.1.5 Stream Isolation Constraint. The transmission of multiple streams via the same queue can cause variations in the arrival times of the frames belonging to different streams. To ensure deterministic arrival of the streams, the offsets of all frames in a stream must be constrained to be less than the offset of the first frame in every other stream. Figure 4 shows an example of the case where stream isolation constraint is applicable. In this example, two streams s_i and s_j are transmitted to the end station v_a forwarding to the same end station v_b. The stream isolation constraint ensures that when the first frame of a stream, s_j in this example, is scheduled for transmission no other frames of other streams, s_i in this example, is scheduled until all frames of s_j are dispatched. In Figure 4, for simplicity of illustration we assumed that the periods of s_i and s_j are the same, hence the hyper period of them are the same as their periods. This constraint is presented in Eq. (5).

$$\begin{aligned}&\forall [V_a,V_b] \in L, \forall s_i^{[V_a,V_b]} \in S, s_j^{[V_a,V_b]} \in S, s_i.p = ST, s_j.p = ST, i \neq j, \\ &f_{i,k}^{[V_a,V_b]} \in s_i^{[V_a,V_b]}, f_{i,l}^{[V_a,V_b]} \in s_i^{[V_a,V_b]}, \\ &\forall \alpha \in [0, \frac{lcm(s_i.T,s_j.T)}{s_i.T} - 1], \forall \beta \in [0, \frac{lcm(s_i.T,s_j.T)}{s_j.T} - 1] : \\ &((f_{j,|s_j|}^{[V_a,V_b]}.\phi \times [V_a,V_b].mt) + f_{j,|s_j|}^{[V_a,V_b]}.Len + (\beta \times s_j.T) + \delta \leq \\ &(f_{i,1}^{[V_x,V_a]}.\phi \times [V_x,V_a].mt) + (\alpha \times s_i.T) + [V_x,V_a].d) \vee \\ &((f_{i,|s_i|}^{[V_a,V_b]}.\phi \times [V_a,V_b].mt) + f_{i,|s_i|}^{[V_a,V_b]}.Len + (\alpha \times s_i.T) + \delta \leq \\ &(f_{j,1}^{[V_y,V_a]}.\phi \times [V_y,V_a].mt) + (\beta \times s_j.T) + [V_y,V_a].d)\end{aligned} \tag{5}$$

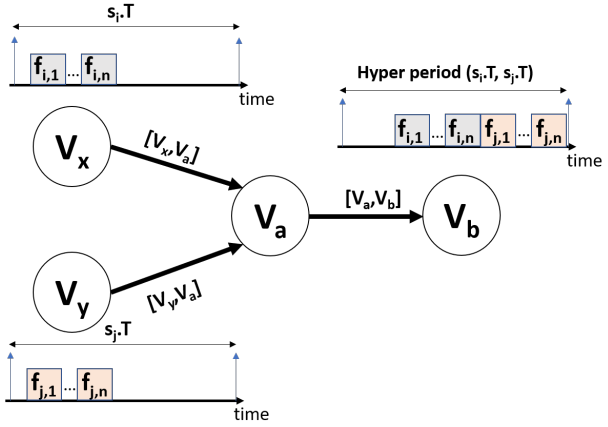


Figure 4: Stream isolation constraint.

4.1.6 Frame Isolation Constraint. In order to relax the stream isolation constraint by allowing frame interleaving between different streams an alternative constraint can be defined. The frame isolation constraint allows interleaving of frames from different streams considering that there are only frames of one stream in the queue at a time. When two source end stations V_x and V_y transmit streams via one or more shared switches within their path, the interleaving of frames in the same queue of the shared switch can cause non-deterministic delays of frames. For example, a stream with a higher period can cause long delays to a shorter period stream in case interleaving of frames occurs between these two streams. The frame isolation constraint, shown in Eq. (6), isolates the transmission of every frame of different streams in the queue. In other words, it guarantees that the offset plus the transmission time of each frame belonging to a stream (s_i transmitted from node V_x) is smaller than or equal to the offset of each frame belonging to another stream (s_j transmitted from node V_y) or vice versa. This scenario is depicted in Figure 5 assuming that both s_i and s_j have the same periods, hence the same hyper period on link $[v_a, v_b]$ with two possible schedules where one of them shows an interleaving of frames from the two streams.

$$\begin{aligned}&\forall [V_a,V_b] \in L, \forall s_i^{[V_a,V_b]} \in S, s_j^{[V_a,V_b]} \in S, s_i.p = ST, s_j.p = ST, i \neq j, \\ &f_{i,k}^{[V_a,V_b]} \in s_i^{[V_a,V_b]}, f_{i,l}^{[V_a,V_b]} \in s_i^{[V_a,V_b]}, \\ &\forall \alpha \in [0, \frac{lcm(s_i.T,s_j.T)}{s_i.T} - 1], \forall \beta \in [0, \frac{lcm(s_i.T,s_j.T)}{s_j.T} - 1] : \\ &((f_{j,l}^{[V_a,V_b]}.\phi \times [V_a,V_b].mt) + f_{j,l}^{[V_a,V_b]}.Len + (\beta \times s_j.T) + \delta \leq \\ &(f_{i,k}^{[V_x,V_a]}.\phi \times [V_x,V_a].mt) + (\alpha \times s_i.T) + [V_x,V_a].d) \vee \\ &((f_{i,k}^{[V_a,V_b]}.\phi \times [V_a,V_b].mt) + f_{i,k}^{[V_a,V_b]}.Len + (\alpha \times s_i.T) + \delta \leq \\ &(f_{j,l}^{[V_y,V_a]}.\phi \times [V_y,V_a].mt) + (\beta \times s_j.T) + [V_y,V_a].d)\end{aligned} \tag{6}$$

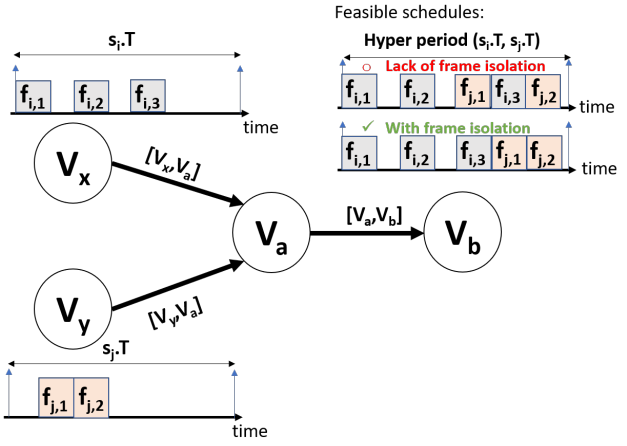


Figure 5: Frame isolation constraint.

4.2 Proposed Constraints

In this section, the problem that motivates the contributions of this paper is presented by an example, shown in Figure 6. In Figure 6(a) time stamps $[0, T_0]$, $[T_1, T_2]$ and $[T_3, T_4]$ are reserved for ST frames. The time slots represented by white boxes in Figure 6(a) indicate the

times when the gates are open for transmission of BE frames. Based on this schedule, ST frames that are packed subsequently within the time slot $[T_1, T_2]$, cause deadline miss for the BE frame, $BE3$. On the other hand, Figure 6(b) shows the sparsification of ST frames within time stamp $[T_1, T_2]$ from the previous example into new scheduled time stamps $[T_1, T_2]$, $[T_3, T_4]$ and $[T_5, T_6]$. Consequently, with a few more gate state changes, the BE frame meets its deadline.

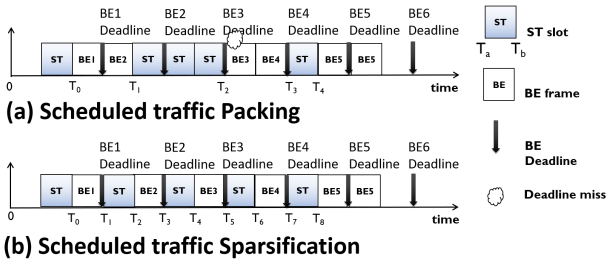


Figure 6: Example of schedule bin-packing and the effect on schedulability of BE traffic.

In order to obtain sparsity in the ST schedules in favor of BE frames, we propose a notion of *slack* after each ST frame transmission. Slack is a time interval after transmission of a ST frame, during which no ST frame can occupy the link's bandwidth. This prevents back-to-back transmission of ST frames [30]. We denote the slack per frame per link by $f_{i,j}^{[v_a,v_b]}.slack$. Figure 7 illustrates the proposed slack property of a frame per link. The goal is to find a feasible ST schedule such that a desired slack per frame per link is obtained. The realization of slacks for all frames leads to implementing spaces between ST frames. The size of slack is defined by the proposed constraints.

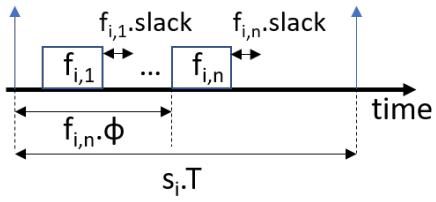


Figure 7: Illustration of the proposed slack property of a frame per link.

4.2.1 Porous Link Constraint. In order to incorporate the slack, the existing link constraints should be modified. The link constraint presented in Eq. (2) only restricted the timing overlap and placement of subsequent frames on the link. The constraint presented in Eq. (2) should be adapted into Eq. (7); the reason for this modification is that the original link constraint only restricted the timing overlap and subsequent placement of frames on the link.

$$\begin{aligned}
&\forall s_i, s_j \in S, s_i.p = ST, s_j.p = ST, \forall [V_a, V_b] \in L, \\
&\forall s_i^{[V_a,V_b]}, s_j^{[V_a,V_b]}, i \neq j, f_{i,k}^{[V_a,V_b]} \in s_i^{[V_a,V_b]}, f_{i,l}^{[V_a,V_b]} \in s_j^{[V_a,V_b]}, \\
&\forall \alpha \in [0, \frac{lcm(s_i.T, s_j.T)}{s_i.T} - 1], \forall \beta \in [0, \frac{lcm(s_i.T, s_j.T)}{s_j.T} - 1] : \\
&(f_{j,l}^{[V_a,V_b]}.\phi + (\beta \times f_{j,l}^{[V_a,V_b]}.T) + f_{j,l}^{[V_a,V_b]}.Len + f_{j,l}^{[V_a,V_b]}.slack \leq \\
&f_{i,k}^{[V_a,V_b]}.\phi + (\alpha \times f_{i,k}^{[V_a,V_b]}.T)) \vee \\
&(f_{i,k}^{[V_a,V_b]}.\phi + (\alpha \times f_{i,k}^{[V_a,V_b]}.T) + f_{i,k}^{[V_a,V_b]}.Len + f_{i,k}^{[V_a,V_b]}.slack \leq \\
&f_{j,l}^{[V_a,V_b]}.\phi + (\beta \times f_{j,l}^{[V_a,V_b]}.T))
\end{aligned} \tag{7}$$

4.2.2 Slack Size Constraint. The slack size constraint asserts the allowed slacks size for each frame scheduled on the link. The slack must be greater than or equal to zero but less than or equal to the difference between the frame period and its transmission time as follows:

$$\begin{aligned}
&\forall s_i \in S, s_i.p = ST, \forall [V_a, V_b] \in L, \forall f_{i,j}^{[V_a,V_b]} \in s_i^{[V_a,V_b]} : \\
&(f_{i,j}^{[v_a,v_b]}.slack \geq 0) \wedge \\
&(f_{i,j}^{[v_a,v_b]}.slack \leq f_{i,j}^{[v_a,v_b]}.T - f_{i,j}^{[v_a,v_b]}.Len)
\end{aligned} \tag{8}$$

The size of slacks on a link must be correlated with the number of ST frames scheduled on the link. Eq. (9) is the summation formulation of the frame slacks on each link. Figure 8 presents an example of the slack spaces after two frames from different streams scheduled on the link $[V_a, V_b]$.

$$\begin{aligned}
&\forall s_i \in S, s_i.p = ST, \forall [V_a, V_b] \in L, \forall f_{i,j}^{[V_a,V_b]} \in s_i^{[V_a,V_b]} : \\
&hopSum^{[v_a,v_b]} = \sum f_{i,j}^{[v_a,v_b]}.slack
\end{aligned} \tag{9}$$

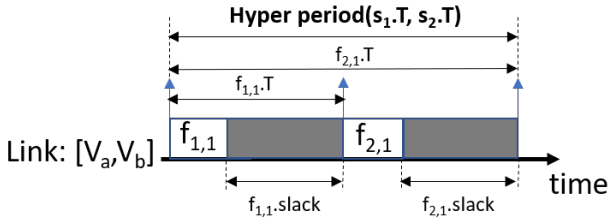


Figure 8: The allowed range for slacks on a link.

4.2.3 Hop Slacks Constraint. In order to bound the total amount of slacks allowed on the link, the constraints in Eq. (10) and Eq. (12) enforce the valid ranges of *hopSum*. The *minPorosity* parameter is considered as the lower bound for *hopSum* summation formulation, as shown in Eq. (10).

$$\begin{aligned}
&\forall s_i \in S, s_i.p = ST, \forall [V_a, V_b] \in L, \forall f_{i,j}^{[V_a,V_b]} \in s_i^{[V_a,V_b]}, \\
&hopSum^{[v_a,v_b]} \geq minPorosity
\end{aligned} \tag{10}$$

where, *minPorosity* can be any value from 0, however, if the value is selected between 0 and the frame size, assuming that the preemption

is disabled, the slack after each frame is not enough for transmission of any BE frames.

Moreover, the sparse spaces on each link need to be specified per link, since the load of BE streams, converged with ST frames, varies on different links. The ST load on the link $[v_a, v_b]$ is specified by $[v_a, v_b].util_{ST}$ and is calculated by the equation:

$$\begin{aligned} &\forall [V_a, V_b] \in L, \forall s_i \in S, s_i.p = ST : \\ &[V_a, V_b].util_{ST} = \sum \frac{s_i^{[v_a, v_b]}.Size}{s_i^{[v_a, v_b]}.T} \end{aligned} \tag{11}$$

Furthermore, the upper bound of the *hopSum* is presented in another constraint to ensure that slacks do not cause the porous link schedule to exceed the hyper period. Eq. (12) ensures that *hopSum* is less than or equal to the total load that is left after the scheduled time for the ST frames on the link.

$$\begin{aligned} &\forall s_i \in S, s_i.p = ST, \forall [V_a, V_b] \in L, \forall f_{i,j}^{[V_a, V_b]} \in s_i^{[V_a, V_b]}, \\ &hopSum^{[v_a, v_b]} \leq lcm(S) \times (1 - [v_a, v_b].util_{ST}) \end{aligned} \tag{12}$$

4.2.4 Equal Slack Constraint. Equal slack is an optional constraint to evenly distribute slacks on a link by adjusting equal slack sizes, as defined by Eq. (13). Note that the optimization time can be shortened by constraining equal-sized slacks due to limiting the options for the ST schedules. As a result, this constraint enables a selection trade-off between the freedom to choose ST schedules, enhancing arrival of periodic BE and optimization time.

$$\begin{aligned} &\forall s_i, s_j \in S, s_i.p = ST, s_j.p = ST, \forall [V_a, V_b] \in L, \\ &\forall f_{j,l}^{[V_a, V_b]} \in s_j^{[V_a, V_b]}, \forall f_{i,k}^{[V_a, V_b]} \in s_i^{[V_a, V_b]}, i \neq j : \\ &f_{i,k}^{[v_a, v_b]}.slack = f_{j,l}^{[v_a, v_b]}.slack \end{aligned} \tag{13}$$

4.3 Objective Functions

Minimizing the offsets of the ST frames' offsets is the objective function that is considered in all the previous works concerning the schedule synthesis, such as the work in [8]. The minimization objective function, given in Eq. (14), generates offsets that pack the ST frames to the beginning of the schedule subject to the generic constraints presented in Section 4.1.

$$\begin{aligned} &minimize \sum_{\forall [v_a, v_b] \in L} \sum_{\forall s_i \in S} \sum_{\forall f_{i,j} \in s_i} f_{i,j}^{[v_a, v_b]}.\phi \\ &\text{subject to} : \{(1), (2), (3), (4), (5) OR (6)\} \end{aligned} \tag{14}$$

The minimization approach can be convenient to secure both schedulability and timeliness of ST traffic. Some of the lower priority streams, including the BE, may be constrained by timing requirements. In order to address these requirements, three new objective functions are proposed as alternatives to the widely used minimization optimization objective function.

4.3.1 Maximization. This optimization objective function still packs ST frames together, but in contrast to the minimization objective function, it schedules the transmission of the ST frames as close as possible to their deadlines. As the number of ST frames increases, the ST reserved time slots on the link also increases. This can increase the maximum end-to-end delay of the BE frames, which in turn, may cause deadline misses. If we maximize the ST schedules, we allow the majority of open bandwidth for BE frames in the beginning of the schedule, increasing the likelihood of transmission of the BE frames before the scheduled transmission of the ST frames. This can improve the schedulability of the BE frames that may have deadlines shorter than those of the ST frames. Eq. (15) defines the maximization objective function subject to the generic constraints presented in Section 4.1.

$$\begin{aligned} &maximize \sum_{\forall [v_a, v_b] \in L} \sum_{\forall s_i \in S} \sum_{\forall f_{i,j} \in s_i} f_{i,j}^{[v_a, v_b]}.\phi \\ &\text{subject to} : \{(1), (2), (3), (4), (5) OR (6)\} \end{aligned} \tag{15}$$

4.3.2 Sparse Schedule. In order to increase the porosity of the ST schedules on each link, we propose sparse objective function by Eq. (16). The sparse objective function adjusts the ST offsets implicitly by maximizing the sum of slacks between subsequent frames, that are scheduled on the same link. As a result, the ST frames are scheduled in a sparse manner to create unequal slacks for the transmission of the lower priority frames. The sparse objective function is subject to the generic constraints as well as the newly proposed constraints.

$$\begin{aligned} &maximize \sum_{[v_a, v_b] \in L} hopSum^{[v_a, v_b]} \\ &\text{subject to} : \{(1), (3), (4), (5) OR (6), (7), (8), (10), (12)\} \end{aligned} \tag{16}$$

4.3.3 Evenly Sparse Schedule. The proposed sparse schedule objective function can be manipulated by an additional constraint, in Eq. (17), to create evenly distributed ST frames, or in other words equally-sized slacks on the links.

$$\begin{aligned} &maximize \sum_{[v_a, v_b] \in L} hopSum^{[v_a, v_b]} \\ &\text{subject to} : \{(1), (3), (4), (5) OR (6), (7), (8), (10), (12), (13)\} \end{aligned} \tag{17}$$

In crux, the minimization and maximization objective functions send ST frames with less number of gate state change, by packing the frames in shared transmission slots. On the other hand, these constraints do not consider slacks between ST frames, thus they cause long queuing times for BE frames either in the beginning or at the end of the hyper period. The sparse and evenly sparse objective functions take advantage of new optimization variables that define slacks after each ST frame on a link. This causes porosity in bandwidth used by the ST frames on each link. Consequently, the BE frames can be fitted in adequate slacks created along the scheduled times for ST frames. The sparse and even sparse objective functions also provide the freedom to squeeze the ST schedule search space by constraining the size of slacks, which is simultaneously beneficial to reduce the optimization time and control the distribution pattern of the slacks.

5 EXPERIMENTAL EVALUATION

In this section, we present the evaluation setup followed by a simulation-based experimental evaluation to show the effects of proposed objective functions on the QoS of the BE frames.

5.1 Evaluation Setup

In order to perform the evaluation, we consider a multi-hop network topology that consists of six end stations. The end stations are connected via two TSN switches as shown in Figure 9.

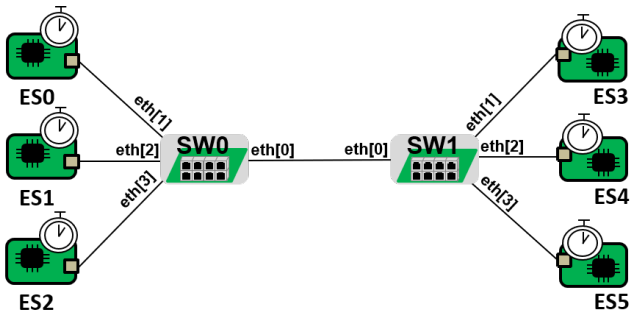


Figure 9: Evaluation setup.

To evaluate the QoS of BE frames we use the NeSTiNg TSN simulation tool [9], which is based on the OMNeT++/INET framework. NeSTiNg facilitates the simulation of networks that are based on the TSN standards. It supports all traffic classes and the TAS mechanism. We developed a plug-in[3] for the NeSTiNg simulation tool that facilitates the insertion of streams, configurations of the gates' states setting and extraction of the simulation results.

In this experiment, we generate random sets of streams, where each stream contains one frame with the maximum size of 1542 Bytes (including the frame header). The source and destination end stations of the streams are selected randomly among the stations shown in the network topology (Figure 9). The streams' periods are selected randomly from the set [200, 500, 1000, 2000, 5000] μs to generate a realistic data set according to the real-world automotive benchmarks [18]. We assume that the network speed is 1 $Gbit/s$ in this example. Using the network speed, the transmission time of each frame with 1542 Bytes of frame size is equal to 12.336 μs. The link macrotick for all links is assumed to be 1 μs and we neglect the link propagation delay. The generated streams are selected among ST and BE classes. Three different scenarios are considered in this experiment, where 10 sets of streams are randomly generated in each scenario. Each set contains a mix of BE and ST streams. The three scenarios are as follows:

- Scenario (1) : 10 random streams with 0.5 probability of ST streams and 0.5 probability of BE streams.
- Scenario (2) : 10 random streams with 0.6 probability of ST streams and 0.4 probability of BE streams.
- Scenario (3) : 10 random streams with 0.8 probability of ST streams and 0.2 probability of BE streams.

To schedule the ST streams we consider the four proposed optimization objectives including: (i) minimization; (ii) maximization; (iii) sparse; and (iv) evenly sparse schedule; In the case of sparse and evenly sparse objective functions, $minPorosity$ is assumed to be zero to make the schedules as flexible as possible. Note that increasing the $minPorosity$ value can reduce the number of feasible schedules, while at the same time it can produce larger slacks for transmission of BE frames. The network is simulated according to the schedules generated by the four different objective functions. The two main metrics, measured during the simulations, include the maximum end-to-end delays and deadline misses of the BE frames.

The objective functions and constraints are implemented in Python and solved by Z3's [32] OMT optimization module. The optimizations and simulations were run on an HP Elite Book 820 running Ubuntu OS 18.04.4 LTS with CPU Core i5, 4 × 2.20 GHz Cores and of 16 GB RAM.

5.2 Results discussion

The maximum end-to-end delay of the generated BE streams are measured during the simulation given that the ST streams are scheduled according to the schedules generated by the proposed objective functions. Figure 10 illustrates the measured maximum end-to-end delays of the BE streams for the three scenarios, i.e, Scenario (1) in Figure 10a, Scenario (2) in Figure 10b, and Scenario (3) in Figure 10c.

Each bar in the graph shows the measured maximum end-to-end delays of all BE streams in the 10 generated sets by highlighting the minimum, average and maximum of these values. For instance, in Scenario (1) 10 sets of streams were generated randomly where each set consists of 10 streams. The chance of the streams becoming ST or BE were 50% in this scenario. Therefore, we have 10 sets with possibly 5 streams per set belong to the BE class. Consequently, the Sparse bar in Figure 10a, for example, shows the maximum end-to-end delays of possibly 50 BE streams, where the average, maximum and minimum values of the measured maximum end-to-end delays among these 50 BE streams are 50.6μs, 98.3μs and 24.5μs, respectively.

As it can be seen in the figure, the minimization objective function gives the worst results compared to the other objective functions with maximum value of 101.2μs in Scenario (1), 102.5μs in Scenario (2) and 90.2μs in Scenario (3). Note that all previous works on the ST scheduling commonly apply the minimization objective function. In the first glance, it can be seen that the maximization objective function outperforms the other objective functions. The main reason is that when the ST schedules are using the time-slots close to their deadlines, more spaces will be available for transmission of possible BE frames. This is in contrast to the minimization objective function that fills the time-slots in the beginning of the ST schedules, leaving spaces for the BE streams after the transmission of ST streams that can potentially result in larger end-to-end delays for the BE frames. However, there are also few extreme cases in which the measured maximum end-to-end delays of BE streams are very high when using the maximization objective function. This is specifically the case in Scenario (2) and Scenario (3). According to the results, the maximization objective function in Scenario (2) resulted in a maximum delay for one BE stream up to 112.2μs and in Scenario (3) up to 212.2μs. These large end-to-end delays are not seen in the sparse and evenly sparse objective functions. The main reason to have these cases with the maximization objective function is that, similar to the minimization objective function, it can pack the ST frames together and prevent transmission of any BE frame for a long duration of time depending on the number of ST frames with close periods. However, the sparse and evenly sparse

[3] https://gitlab.com/abbelini/TSN-plugin

Table 1: Number of deadline misses.

	Objective functions	Total deadline misses
Scenario (1)	Maximization	0
	Minimization	2
	Evenly sparse	1
	Sparse	1
Scenario (2)	Maximization	99
	Minimization	102
	Evenly sparse	2
	Sparse	1
Scenario (3)	Maximization	0
	Minimization	0
	Evenly sparse	0
	Sparse	0

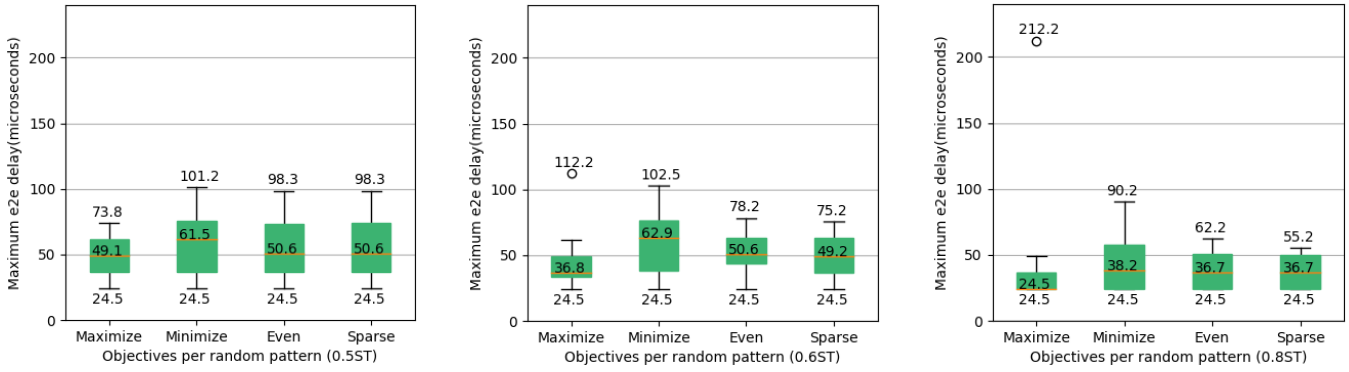
objective functions can generate porosity in the ST schedules that can be sufficiently used for the BE frames' transmission preventing the long blocking time by the ST frames. Therefore, although on average, the maximization objective function performs better than the rest, the sparse and evenly sparse objective functions have their own benefits in various scenarios, e.g., the one discussed above.

We also measured the deadline misses of BE streams in the generated scenarios. We consider implicit deadlines, i.e., the deadline for each BE stream is assumed to be equal to its period. Table 1 shows the total deadline misses among all generated BE streams within each scenario and with each objective function. It can be clearly observed that the minimization objective function can produce more deadline misses compared to the other objective functions, while the sparse and evenly sparse objective functions resulted in fewer deadline misses for the BE streams. For instance, in Scenario (2) the number of deadline misses with minimization and maximization objective functions are 102 and 99, respectively, whereas it is 1 and 2 for sparse and evenly sparse objective functions, respectively. Scenario (3) shows no deadline misses, which can be because of less number of generated BE streams, i.e., 20% chance for the generated streams to be in BE class.

Although the time that it takes to find a feasible and optimized schedule depends on the implementation and the utilized solver, we measured this time in all the experiments. As mentioned before, we used Z3 SMT/OMT solver for this purpose. The average times for each scenario and objective function are shown in Figure 11. An interesting observation is that both sparse and evenly sparse objective functions are much faster to give an optimized ST schedule compared to both minimization and maximization objective functions. For instance, in Scenario (3) the amount of time that it takes to deliver an ST schedule using the sparse objective function in average was 4.75 seconds. For the same scenario, the amount of time that it takes to deliver an ST schedule using the minimization and maximization objective functions were 314.34 seconds and 1153.52 seconds, respectively.

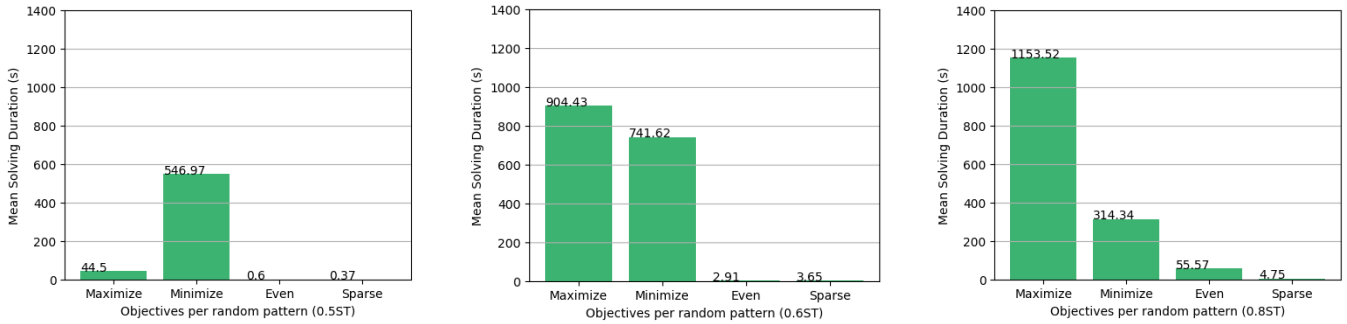
6 CONCLUSION AND FUTURE WORKS

In this paper, we proposed a set of constraints and objective functions to schedule ST traffic in TSN networks considering the QoS of the BE traffic. We argued that in many industrial applications one of the main challenges is to map the legacy traffic into the TSN traffic classes and often the soft real-time traffic are mapped into the BE class. Therefore, a required level of QoS for the BE traffic should be obtained while ensuring a feasible schedule for the hard real-time ST traffic. The solutions proposed in this paper generate feasible schedules for the ST traffic and at the same time significantly reduce the end-to-end delays and the number of deadline misses for the BE traffic. Using a set of experiments, based on the NeSTiNg TSN simulation tool, we showed that the proposed objective functions outperform the state-of-the-art ST scheduling solutions that focus on minimizing the offsets of the ST traffic. We also showed that the feasible ST schedules can be obtained much faster with the proposed solution compared to the state-of-the-art solutions. Although the main focus of the solutions proposed in this paper was on the QoS of BE traffic, we believe that it can also affect positively on the performance of classes A and B. However, a deeper investigation remains for the future as the NeSTiNg simulation tool currently does not support the TAS and CBS mechanisms at the same time. We also plan to investigate the effect of enabling preemption on the quality of service of the BE traffic. Another future research



(a) Scenario (1). **(b) Scenario (2).** **(c) Scenario (3).**

Figure 10: Maximum end-to-end delays of the BE streams under various objective functions per traffic distribution case.



(a) Scenario (1). **(b) Scenario (2).** **(c) Scenario (3).**

Figure 11: Duration of schedule synthesis for each distribution pattern per objective function.

direction is to develop schedulability analysis for the BE traffic to analytically show the performance of the proposed solutions.

ACKNOWLEDGMENTS

The work in this paper is supported by the Swedish Governmental Agency for Innovation Systems (VINNOVA) through the projects DESTINE and PROVIDENT, and by the Swedish Knowledge Foundation (KKS) through the projects DPAC, HERO & FIESTA. The authors thank all industrial partners, especially Arcticus Systems, Volvo CE, and HIAB Sweden.

REFERENCES

[1] Nikolaj Bjørner, Anh-Dung Phan, and Lars Fleckenstein. 2015. *νZ - An Optimizing SMT Solver.* In *Tools and Algorithms for the Construction and Analysis of Systems*, Christel Baier and Cesare Tinelli (Eds.). Springer Berlin Heidelberg.

[2] S. S. Craciunas and R. S. Oliver. 2016. Combined Task-and Network-level Scheduling for Distributed Time-triggered Systems. *Real-Time Systems.*

[3] S. S. Craciunas and R. S. Oliver. 2017. An Overview of Scheduling Mechanisms for Time-sensitive Networks. *Proceedings of the Real-time Summer School.*

[4] S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner. 2016. Scheduling Real-Time Communication in IEEE 802.1Qbv Time Sensitive Networks. In *Proceedings of the 24th Intl. Conference on Real-Time Networks and Systems.*

[5] S. S Craciunas, R. S. Oliver, and Wilfried Steiner. 2017. Formal scheduling constraints for time-sensitive networks. *arXiv preprint arXiv:1712.02246* (2017).

[6] S. S. Craciunas, R. S. Oliver, and W. Steiner. 2018. Demo Abstract: Slate XNS–An Online Management Tool for Deterministic TSN Networks. In *2018 IEEE Real-Time and Embedded Technology and Applications Symposium.*

[7] A. C. T. dos Santos, B. Schneider, and V. Nigam. 2019. TSNSCHED: Automated Schedule Generation for Time Sensitive Networking. In *2019 Formal Methods in Computer Aided Design.* IEEE.

[8] F. Dürr and N. Nayak. 2016. No-wait Packet Scheduling for IEEE Time-sensitive Networks (TSN). In *Proceedings of the 24th Intl. Conference on Real-Time Networks and Systems.*

[9] J. Falk, D. Hellmanns, B. Carabelli, N. Nayak, F. Dürr, S. Kehrer, and K. Rothermel. 2019. NeSTiNg: Simulating IEEE Time-sensitive Networking (TSN) in OMNeT++. In *Proceedings of the 2019 Intl. Conference on Networked Systems.* IEEE.

[10] M. H. Farzaneh. 2019. *A Modeling Framework to Facilitate Schedule Synthesis of Time-Sensitive Networking.* Ph.D. Dissertation. Technische Universität München.

[11] M. H. Farzaneh and A. C. Knoll. 2016. An ontology-based Plug-and-Play approach for in-vehicle Time-Sensitive Networking (TSN). In *7th IEEE Annual Information Technology, Electronics and Mobile Communication Conference.*

[12] M. H. Farzaneh and A. C. Knoll. 2017. Time-sensitive networking (TSN): An experimental setup. In *IEEE Vehicular Networking Conference.*

[13] M. H. Farzaneh, S. Kugele, and A. C. Knoll. 2017. A graphical modeling tool supporting automated schedule synthesis for time-sensitive networking. In *2017 22nd IEEE Intl. Conference on Emerging Technologies and Factory Automation.*

[14] M. H. Farzaneh, S. Shafaei, and A. C. Knoll. 2016. Formally verifiable modeling of in-vehicle time-sensitive networks (TSN) based on logic programming. In *2016 IEEE Vehicular Networking Conference.*

[15] V. Gavriluț, L. Zhao, M. L. Raagaard, and P. Pop. 2018. AVB-Aware Routing and Scheduling of Time-Triggered Traffic for TSN. *IEEE Access* (2018).

[16] IEEE. 2018. IEEE Std. 802.1Q, IEEE Standard for Local and metropolitan area networks, Bridges and Bridged Networks. (2018).

[17] IEEE. 2020. IEEE Std. 802.1AS, IEEE Standard for Local and Metropolitan Area Networks-Timing and Synchronization for Time-Sensitive Applications. (2020).

[18] S. Kramer, D. Ziegenbein, and A. Hamann. 2015. Real World Automotive Benchmarks for Free. In *6th Intl. Workshop on Analysis Tools and Methodologies for Embedded and Real-Time Systems.*

[19] L. Lo Bello, R. Mariani, S. Mubeen, and S. Saponara. 2019. Recent Advances and Trends in On-Board Embedded and Networked Automotive Systems. *IEEE Transactions on Industrial Informatics* 15, 2 (2019).

[20] L. Lo Bello and W. Steiner. 2019. A Perspective on IEEE Time-Sensitive Networking for Industrial Communication and Automation Systems. *Proc. IEEE* (2019).

[21] Y. Mi, J. Qu, J. Zhang, and M. Yao. 2020. A Scheduling Algorithm of Maximize the Number of Porosity for the Time-Triggered DIMA System. In *2020 IEEE 3rd Intl. Conference on Electronics Technology.*

[22] M. Pahlevan and R. Obermaisser. 2018. Genetic Algorithm for Scheduling Time-Triggered Traffic in Time-Sensitive Networks. In *2018 IEEE 23rd Intl. Conference on Emerging Technologies and Factory Automation.*

[23] M. Pahlevan, J. Schmeck, and R. Obermaisser. 2019. Evaluation of TSN Dynamic Configuration Model for Safety-Critical Applications. In *2019 IEEE Intl Conf on Parallel Distributed Processing with Applications, Big Data Cloud Computing, Sustainable Computing Communications, Social Computing Networking (ISPA/BDCloud/SocialCom/SustainCom).* 566–571.

[24] M. Pahlevan, N. Tabassam, and R. Obermaisser. 2019. Heuristic List Scheduler for Time Triggered Traffic in Time Sensitive Networks. *SIGBED Rev.* (Feb. 2019).

[25] F. Pozo, G. Rodriguez-Navas, and H. Hansson. 2018. Schedule Reparability: Enhancing Time-Triggered Network Recovery Upon Link Failures. In *2018 IEEE 24th Intl. Conference on Embedded and Real-Time Computing Systems and Applications.*

[26] N. Reusch, L. Zhao, S. S. Craciunas, and P. Pop. 2020. Window-Based Schedule Synthesis for Industrial IEEE 802.1Qbv TSN Networks. In *2020 16th IEEE Intl. Conference on Factory Communication Systems.*

[27] R. S. Oliver, S. S. Craciunas, and W. Steiner. 2018. IEEE 802.1Qbv Gate Control List Synthesis Using Array Theory Encoding. In *2018 IEEE Real-Time and Embedded Technology and Applications Symposium.*

[28] B. Schneider. 2019. Automatic Network Configuration for Real-Time, Distributed Industrial Automation Systems. In *2019 ACM/IEEE 22nd Intl. Conference on Model Driven Engineering Languages and Systems Companion.*

[29] Wilfried Steiner. 2010. An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks. In *31st IEEE Real-Time Systems Symposium.*

[30] W. Steiner. 2011. Synthesis of Static Communication Schedules for Mixed-Criticality Systems. In *2011 14th IEEE Intl. Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops.*

[31] TSNSched. 2020. *Automated Schedule Generation for TSN networks.* https://github.com/ACassimiro/TSNsched

[32] Z3Py. 2020. *Z3 is a theorem prover from Microsoft Research.* https://pypi.org/project/z3-solver/

[33] L. Zhao, P. Pop, and S. S. Craciunas. 2018. Worst-Case Latency Analysis for IEEE 802.1Qbv Time Sensitive Networks Using Network Calculus. *IEEE Access* (2018).