Scheduling Real-Time Communication in IEEE 802.1Qbv Time Sensitive Networks*

Silviu S. Craciunas Ramon Serna Oliver Martin Chmelík Wilfried Steiner
TTTech Computertechnik AG
Schönbrunner Straße 7
1040 Vienna, Austria
{silviu.craciunas, ramon.serna.oliver, martin.chmelik, wilfried.steiner}@tttech.com

ABSTRACT

The enhancements being developed by the Time-Sensitive Networking Task Group as part of IEEE 802.1 emerge as the future of real-time communication over Ethernet networks for automotive and industrial application domains. In particular IEEE 802.1Qbv is key to enabling timeliness guarantees via so-called *time-aware shapers*. In this paper, we address the computation of fully deterministic schedules for 802.1Qbv-compliant multi-hop switched networks. We identify and analyze key functional parameters affecting the deterministic behaviour of real-time communication under 802.1Qbv and, based on a generalized configuration of these parameters, derive the required constraints for computing offline schedules guaranteeing low and bounded jitter and deterministic end-to-end latency for critical communication flows. Furthermore, we discuss several optimization directions and concrete configurations exposing trade-offs against the required computation time. We also show the performance of our approach via synthetic network workloads on top of different network configurations.

1. INTRODUCTION

Ethernet has evolved to be the standard open communication mechanism for a wide range of application domains originally not bound to strict timing requirements. Typical design and performance criteria for standard Ethernet deployments are related to best-effort communication, in which maximizing throughput and minimizing average delays are primary objectives. In the real-time domain, safety-critical timing aspects have been introduced by means of technologies like TTEthernet (SAE AS6802 [21, 34]), PROFINET, and EtherCAT [27] among others. Recently, the convergent needs of the industrial automation and automotive domains are pushing for the standardization of solutions enabling mixed-criticality communication allowing real-time and best-effort traffic to coexist simultaneously within IEEE 802 networks. Quality of service, as introduced in the IEEE 802.1BA Audio/Video Bridging (AVB) standard, has raised significant interest within the industrial and automotive domains, but falls short of the industry expectations for hard real-time applications demanding reliable and fully deterministic communication [2, 25, 32]. Enhancing the features of AVB with fault-tolerant mechanism allowing strict timing guarantees with low latency and jitter is desirable to support real-time applications over Ethernet.

The IEEE 802.1 Time Sensitive Networking (TSN) task group [20] is in the process of standardizing time-sensitive capabilities over IEEE 802 networks by defining a whole range of 802.1 sub-standards, covering aspects that range from clock synchronization and frame preemption to redundancy management and scheduled traffic enhancements. Key enablers of real-time communication in TSN networks are a network-wide clock reference with bounded precision, known and bounded network latencies (e.g. switch forwarding times, link delays), as well as isolation of critical and non-critical traffic classes through a global communication schedule. IEEE 802.1ASrev [20] defines the basis for a network-wide time-synchronization protocol, achieving a global network clock and basic fault-tolerance by means of a re-election mechanism of the clock master (also called grandmaster) in case of failures. Time sensitive communication takes place via so-called time-sensitive streams (or flows), where a stream is defined by a sender (or *talker*) and one or multiple receivers (or *listeners*). Individual messages belonging to a flow are defined using the VLAN identifier (VID) of VLAN-tagged frames. Streams can have different priorities defining their traffic class, given by the 3 priority bits of the priority code point (PCP) of the IEEE 802.1Q header. IEEE 802.1Qbv [19] defines a time-based shaper functionality enabling time-triggered communication [22] at the egress ports. A time-aware shaper is essentially a gate enabling or disabling the transmission of frames for a queue following the specification of a periodic schedule.

Traditionally, worst-case end-to-end communication latencies in non-scheduled networks (i.e., using priority queues) have been analyzed through methods like network calculus [12, 14] or the trajectory approach [4]. In such approaches, the end-to-end latency of flows is computed based on traffic priorities and arrival patterns making compositional system design and isolation of time behaviour very difficult. With the introduction of 802.1Qbv, the comple-

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 700665 (CITADEL).

RTNS '16, October 19-21, 2016, Brest, France

DOI: http://dx.doi.org/10.1145/2997465.2997470

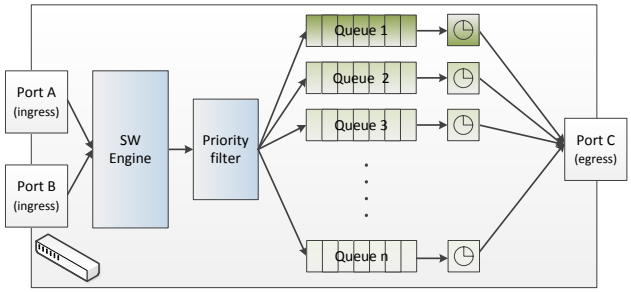


Figure 1: *Simplified view of a 802.1Qbv-capable switch.*

mentary approach, which we present in this paper, is now also supported, where correct schedules with respect to individual flow requirements are created offline. The major benefit of the 802.1Qbv extension is that it allows temporal isolation and compositional system design when flows are scheduled end-to-end.

In our previous work [7, 8, 33], we have addressed several scheduling problems in TTEthernet networks. However, the particularities of TSN demand different solutions than those employed in the past. In this paper, we address the scheduling problem arising from the IEEE 802.1Qbv extension on multi-hop fully switched TSN networks, presenting a method for computing static schedules for 802.1Qbv-capable network devices using Satisfiability Modulo Theories (SMT) and, alternatively, Optimization Modulo Theories (OMT) solvers. We first identify key functional parameters affecting the behaviour of TSN networks and extract a generalized configuration of these parameters for hard real-time networks (Section 2). In Section 3 we describe the formalization of the network model used throughout the paper. Based on the formalization and the generic configuration, we adapt a subset of constraints general to deterministic Ethernet from previous work and, additionally, construct TSN-specific constraints that correctly formulate the real-time behaviour of TSN streams (Section 4). In Sections 5 and 6 we discuss the implication of different configurations on the scheduling problem and present implementation details of our method using SMT and OMT solvers together with a discussion on optimization objectives. After the experimental evaluation in Section 7, we survey related work in Section 8 and conclude the paper in Section 9.

2. 802.1Qbv FUNCTIONAL PARAMETERS

Figure 1 depicts a logical representation of a simplified 802.1Qbv-capable switch with three ports. The figure depicts a scenario in which network traffic traverses the switch from an ingress port to an egress port. Internally, the ports are connected to a filter distributing incoming frames to the associated queue of the egress port based on classification criteria like e.g. the priority code point (PCP) of the IEEE 802.1Q header. On regular switches with multiple ports, incoming traffic will be directed to the corresponding egress port by the switching engine. Every port will have a similar logical composition on egress as the one depicted, including a series of logical queues buffering the respective frames until their transmission. In the most generic representation, each 802.1Qbv queue has a timed gate associated to it enabling or disabling the transmission of frames according to a predefined static schedule. Hence, scheduled events determine at which time instants a queue is opened and traffic is forwarded to the egress port and at which time instants the queue is closed such that any pending traffic remains buffered. If multiple queues are opened at the same time, the queue priority determines which of the queues is allowed to forward frames.

We identify two key parameters that affect the timing properties of network flows in 802.1Qbv-capable networks, namely *Device capabilities* and *Queue configuration*.

One important characteristic is the capability of devices, namely, if they are scheduled or not. If the end-systems are not scheduled but switches are, frames from the end-nodes may arrive in any order to the switches. Therefore, the first switch on the route has to act as a synchronization gate. This synchronization is possible if there are enough queues to isolate flows arriving simultaneously, allowing thus each of them to be scheduled deterministically. Complementary, end-systems may be scheduled but switches are not. In such a case, switches act as delay elements but cannot control the time of forwarding frames. If end-systems and switches are scheduled, the scheduler can plan when to send individual frames of flows such that they arrive with a known order at the first switch and switches can control forwarding times. We denote a network configuration ($G(E)$) where only end-system are scheduled with $\mathcal{V}_e$ and one where only switches are scheduled with $\mathcal{V}_s$. We denote fully scheduled networks, i.e., all devices are scheduled, with configuration $\mathcal{V}_{e+s}$. Here we focus on deterministic networks with scheduled end-systems and switches and leave the other corner-cases for a more in-depth analysis in future work.

The number of queues per egress port, which translates to the number of different priorities that can be handled, affects the amount of traffic and the real-time properties to be guaranteed. Another important aspect is how the queues operate. Queues may follow a strict priority policy, meaning that the gate is always opened and the arbitration is purely based on their assigned priority. Using the time-gates, however, queues can be scheduled following a time-triggered (TT) paradigm, where the priority is overruled by the schedule, i.e., the schedule of the timed-gate defines the servicing order, not the priority. If queues are scheduled following a mutually exclusive pattern, i.e., no two queue gates are opened at the same time, the result is a fully deterministic forwarding policy. In this paper we focus on creating schedules for the timed-gates of scheduled queues. We denote the queue configuration by the tuple $G(Q) = \langle \aleph, \aleph_{tt}, \aleph_{prio} \rangle$, where $\aleph$ is the total number of queues per port, $\aleph_{tt}$ is the number of queues operating as scheduled (TT) queues, and $\aleph_{prio}$ is the remaining number of priority queues. Without loss of generality we assume that in the system configuration, the scheduled queues always have the highest priorities of all queues, including the remaining priority queues.

Having different queue types opens up the possibility to have two different types of traffic, namely scheduled and non-scheduled traffic. Scheduled traffic is assigned to the scheduled queues in order to guarantee latency and jitter requirements. Non-scheduled traffic is isolated from scheduled traffic within the priority queues of a device. Within the priority queues, non-scheduled flows may interfere with each other, but not with scheduled flows (since the associated scheduled queues always have higher priority). For the non-scheduled traffic, worst-case end-to-end latencies in the presence of interference have been analyzed through meth-

ods like network calculus [14] or the trajectory approach [4] for AFDX networks while recently, similar analysis methods have been employed for AVB networks [12].

Scheduled traffic is defined as having requirements on the maximum end-to-end latency as well as minimal jitter constraints, i.e., each periodic instance of the communication flow has to arrive at the same instant in time. In real-time systems, this traffic is said to have high criticality. Non-scheduled traffic cannot be guaranteed with minimal jitter and may or may not have requirements on the end-to-end latency. In this paper we address the scheduling of high-criticality traffic for fully-scheduled networks consisting of 802.1Qbv devices with $\aleph_{tt} \geq 1$.

A system configuration, composed of the 2 parameters presented above, is defined by the tuple $\langle G(E), G(Q)\rangle$. In Sections 3 and 4 we introduce the 802.1Qbv-specific network and traffic model and define scheduling constraints for high-criticality traffic, respectively, assuming generic parameter configurations for critical communication, namely $\{\mathcal{V}_{e+s}, \langle n, m, n-m\rangle\}$. Our constraints generate schedules for the m queues while ensuring that critical flows meet their latency and jitter requirements. Later, in Section 6, we discuss several scenarios also including priority queues and the resulting trade-offs based on specific system configurations.

3. NETWORK AND TRAFFIC MODEL

Our work targets multi-hop layer 2 switched Ethernet networks over *full-duplex* multi-speed physical links. We model the network, similar to [8] and [33], as a directed graph $G(\mathcal{V}, \mathcal{L})$, where the nodes (switches and end-systems) are the set of graph vertices ($\mathcal{V}$) and the links between nodes are represented through the graph edges ($\mathcal{L} \subseteq \mathcal{V}\times\mathcal{V}$). A full-duplex physical link between nodes $v_a \in \mathcal{V}$ and $v_b \in \mathcal{V}$ results in two directional logical links, each denoted by an ordered tuple, namely $[v_a, v_b] \in \mathcal{L}$ and $[v_b, v_a] \in \mathcal{L}$, respectively. Maintaining the notation from [8], a physical link $[v_a, v_b]$ is characterized by the tuple $\langle [v_a, v_b].s, [v_a, v_b].d, [v_a, v_b].mt, [v_a, v_b].c\rangle$, where $[v_a, v_b].s$ is the speed of the link, $[v_a, v_b].d$ is the propagation delay on the medium, and $[v_a, v_b].mt$ is the macrotick of the link. We extend the definition with a new parameter, $[v_a, v_b].c$, which represents the number of available queues in the device. Since any given egress port is connected to at most one link, we establish an equivalence between the port and its associated link. Therefore, we refer to $[v_a, v_b].c$ as the number of egress queues[1] present in the respective port as a property of the link. We derive the transmission time of a frame between two nodes through the link speed (or bit rate) which represents the transfer rate over the physical network cable. For e.g. on a 1Gbit/sec link an MTU-sized IEEE 802.1Q Ethernet frame of 1542 bytes would have a transmission time of $12.336\mu sec$. The propagation delay is additional delay on frame arrival resulting from the delay on the physical medium and the link length. The macrotick is a design property either given by device constraints or system-level design specifications. It specifies the length of a discrete time unit defining the granularity of time events for the given link.

A stream, or flow, is a periodic multicast data transmission from one sender (talker) to one or multiple receivers (listeners). Without loss of generality, we choose to simplify the notation by restricting the number of receivers to one. Enhancing the model to include multicast flows is a trivial extension without implications on the validity of the presented method. We denote the input set of flows by $\mathcal{S}$. Similar to [33], a flow $s_i \in \mathcal{S}$ from sender node v_a to receiver node v_b, routed through the intermediary nodes (i.e. switches) $v_1, v_2, \ldots, v_{n-1}, v_n$ is expressed as $s_i = [[v_a, v_1], [v_1, v_2], \ldots, [v_{n-1}, v_n], [v_n, v_b]]$. A flow is characterized by the tuple $\langle s_i.e2e, s_i.L, s_i.T\rangle$, denoting the maximum allowed end-to-end latency, the data size in bytes, and the period of the flow, respectively.

In the model differentiate between a flow and the instance of a flow on a link. An instance of a flow $s_i \in \mathcal{S}$ routed through link $[v_a, v_b] \in \mathcal{L}$ is denoted by $s_i^{[v_a,v_b]}$. We denote the queue for the given flow instance in the egress port as a variable $s_i^{[v_a,v_b]}.p$. Note that the queue variable is synonymous to the priority of the flow within the egress port of the respective device. Additionally, we draw an equivalence between the egress port and its respective transmission link. It is important to note that the queue assigned to flow instances of the same flow may or may not change along the subsequent egress ports[2].

Since the data size of a flow is allowed to exceed the Ethernet MTU size in TSN, each flow instance is associated with a set of frames, each with size less than or equal to the MTU size. Maintaining our notation from [8], we denote the set of frames $f_{i,j}^{[v_a,v_b]}$ of a flow instance $s_i^{[v_a,v_b]}$ by $\mathcal{F}_i^{[v_a,v_b]}$. Additionally, the first and last frame of the set, ordered by the schedule offset on the link, are expressed as $\mathcal{F}_i^{[v_a,v_b]}$ with $f_{i,1}^{[v_a,v_b]}$ and $last(\mathcal{F}_i^{[v_a,v_b]})$, respectively. A frame $f_{i,j}^{[v_a,v_b]} \in \mathcal{F}_i^{[v_a,v_b]}$ is defined by the tuple

$$\langle f_{i,j}^{[v_a,v_b]}.\phi, f_{i,j}^{[v_a,v_b]}.T, f_{i,j}^{[v_a,v_b]}.L\rangle,$$

where $f_{i,j}^{[v_a,v_b]}.\phi \in [0, f_{i,j}^{[v_a,v_b]}.T]$ is the offset in macroticks of the frame on link $[v_a, v_b]$, $f_{i,j}^{[v_a,v_b]}.T = \lceil \frac{s_i.T}{[v_a,v_b].mt} \rceil$ is the period of the flow scaled to the link macrotick, and $f_{i,j}^{[v_a,v_b]}.L = \lceil \frac{L_i \times [v_a,v_b].s}{[v_a,v_b].mt} \rceil$ is the transmission duration of the frame also scaled to the link macrotick [8].

4. SCHEDULING CONSTRAINTS

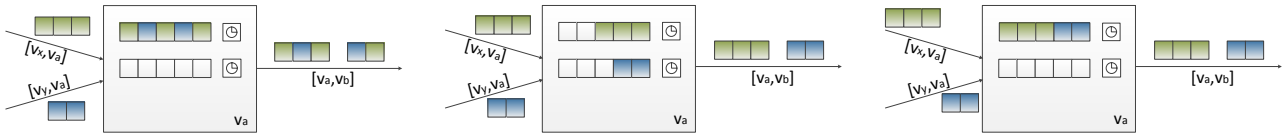
In order compute a schedule for the timed gates of the scheduled queues we define scheduling constraints for the frame offset variables (ϕ) and the flow instance queue variables (p), such that the correct temporal behaviour on the scheduled communication is guaranteed. We differentiate between basic constraints for deterministic Ethernet (Section 4.1) and 802.1Qbv specific constraints (Section 4.2) formalising the specific behaviour of time sensitive flows.

4.1 Basic Deterministic Ethernet Constraints

In [33], a standard TTEthernet model is used where periodic flows are limited to one single frame. In [8], the model is enhanced allowing combined scheduling of preemptive tasks and network flows in a TTEthernet network. The frame description and model from [8], where tasks are modelled as sets of frames, fits well into the multi-frame flow model

[1]The proposed value for this parameter is 8 according to 802.1Qbv [19], however, we do not put any limitation on this number for the scheduling algorithm.

[2]Different queue assignment along multi-hop routes can occur by means of e.g. VLAN re-tagging.



(a) Interleaving flows in egress queue. (b) Isolation of flows in different queues. (c) Isolation of flow arrival.

Figure 2: *Flow interleaving and isolation within an egress port of a 802.1Qbv switch.*

that we discuss here. Hence, in this section we briefly reiterate the most important constraints from [8], adapted and changed to fit our model, and refer the reader to [8, 33] for a more complete and detailed list of scheduling constraints for deterministic Ethernet.

Frame Constraint. The frame offset of any frame scheduled in the network has to be greater than or equal to 0. Additionally, the entire transmission window (offset plus frame duration) has to fit within the frame period. Hence, we have the condition adapted from [8]

$$\forall s_i \in \mathcal{S}, \forall [v_a, v_b] \in s_i, \forall f_{i,j}^{[v_a,v_b]} \in \mathcal{F}_i^{[v_a,v_b]} : \left(f_{i,j}^{[v_a,v_b]}.\phi \geq 0\right) \wedge \left(f_{i,j}^{[v_a,v_b]}.\phi \leq f_{i,j}^{[v_a,v_b]}.T - f_{i,j}^{[v_a,v_b]}.L\right).$$

Link Constraint. No two frames that are routed through the same physical link in the network can overlap in the time domain. The constraint adapted from [8] is as follows

$$\begin{aligned} &\forall [v_a, v_b] \in \mathcal{L}, \forall \mathcal{F}_i^{[v_a,v_b]}, \mathcal{F}_j^{[v_a,v_b]}, i \neq j \\ &\forall f_{i,k}^{[v_a,v_b]} \in \mathcal{F}_i^{[v_a,v_b]}, \forall f_{j,l}^{[v_a,v_b]} \in \mathcal{F}_j^{[v_a,v_b]}, \\ &\forall \alpha \in [0, hp_i^j/s_i.T - 1], \forall \beta \in [0, hp_i^j/s_j.T - 1] : \\ &\Big(f_{i,k}^{[v_a,v_b]}.\phi + \alpha \times f_{i,k}^{[v_a,v_b]}.T \geq \\ &f_{j,l}^{[v_a,v_b]}.\phi + \beta \times f_{j,l}^{[v_a,v_b]}.T + f_{j,l}^{[v_a,v_b]}.L\Big) \vee \\ &\Big(f_{j,l}^{[v_a,v_b]}.\phi + \beta \times f_{j,l}^{[v_a,v_b]}.T \geq \\ &f_{i,k}^{[v_a,v_b]}.\phi + \alpha \times f_{i,k}^{[v_a,v_b]}.T + f_{i,k}^{[v_a,v_b]}.L\Big), \end{aligned} \tag{1}$$

where $hp_i^j = lcm(s_i.T, s_j.T)$ is the hyperperiod of s_i and s_j.

Flow Transmission Constraint. The propagation of frames of a flow must follow the sequential order along the routed path of the flow. The network precision, denoted with δ, represents the worst-case difference between the local clocks of any two synchronized (e.g. via the IEEE 802.1AS [20] time-synchronization protocol) devices. Please note that in the TTEthernet synchronization protocol [35] the synchronization frames typically have the highest priority and introduce a shuffling delay to the critical frames. The IEEE 802.1AS protocol also requires synchronization frames that travel on the same physical links as the critical frames but they can have a lower priority than the critical traffic and hence do not introduce a shuffling delay.

$$\begin{aligned} &\forall s_i \in \mathcal{S}, \forall [v_a, v_x], [v_x, v_b] \in s_i, \\ &\forall f_{i,j}^{[v_a,v_x]} \in \mathcal{F}_i^{[v_a,v_x]}, \forall f_{i,j}^{[v_x,v_b]} \in \mathcal{F}_i^{[v_x,v_b]} : \\ &f_{i,j}^{[v_x,v_b]}.\phi \times [v_x, v_b].mt - [v_a, v_x].d - \delta \geq \\ &(f_{i,j}^{[v_a,v_x]}.\phi + f_{i,j}^{[v_a,v_x]}.L) \times [v_a, v_x].mt. \end{aligned} \tag{2}$$

The constraint imposes that a frame can only be scheduled on a subsequent link $[v_x, v_b]$ after the complete reception on the previous link $[v_a, v_x]$, including the propagation delay of the respective link ($[v_a, v_x].d$). Note that, as opposed to [8], the constraint is set to individual frames rather than the complete flow instance, to allow initiating forwarding as soon as the first frame is fully received without requiring to buffer the entire flow.

End-to-End Constraint. The maximum end-to-end latency constraint specifies that the difference between the arrival and sending time of a flow has to be less than or equal to the specified maximum. We denote the sending link of flow s_i with $src(s_i)$ and the last link before the receiving node with $dest(s_i)$.

$$\begin{aligned} &\forall s_i \in \mathcal{S} : src(s_i).mt \times f_{i,1}^{src(s_i)}.\phi + s_i.e2e \geq \\ &dest(s_i).mt \times (last(\mathcal{F}_i^{dest(s_i)}).\phi + last(\mathcal{F}_i^{dest(s_i)}).L). \end{aligned} \tag{3}$$

4.2 802.1Qbv Constraints

In TTEthernet a static schedule is provided per device port and defines the temporal behaviour of all frames of the scheduled flows whereas in 802.1Qbv the schedule is specified for an egress queue, affecting multiple flows with the same traffic class (priority). Moreover, a TTEthernet stream is defined as one frame of at most MTU size whereas flow in 802.1Qbv can be composed of multiple frames. Hence, 802.1Qbv-specific constraints need to be specified. We elaborate on these constraints using a simplified example, depicted in Figure 2, where two flows arriving at a switch from different sources are forwarded via the same egress port, discussing both the case when they are queued in the same scheduled queue and the case when they use different scheduled queues.

Egress Interleaving. In the first scenario, depicted in Figure 2(a), the two flows arrive from different links at roughly the same time. Regardless of the synchronization protocol used, there will always be an amount of synchronization error between individual devices that may lead to alternating arrival order of frames during runtime. Therefore, the order in which the individual frames are placed in the scheduled queue at runtime is non-deterministic. It is important to note that the schedule controls the timing of the events on the egress port, not the order of frames in the queue. If the timed gate of the respective queue is opened first for the time interval needed to transmit 2 frames and some time later for 3 frames, as depicted in the example, any combination of the respective frames of both flows may occur on the egress port. Hence, the timely behaviour of the two flows may oscillate each periodic instance accumulating jitter for the overall end-to-end transmission. In order to guarantee that the given end-to-end latency is always fulfilled, the worst-case delay introduced on each egress port

along the flow path must be considered in the end-to-end latency constraint for the flow. However, the worst-case jitter that accumulates with each hop along the route of the flow is not desirable in hard real-time systems. Additionally, the non-determinism introduced by the interleaving of individual frames weakens the temporal isolation of flows since a change in one flow may affect other flows by adding interference delay and jitter.

One solution to avoid this delay and jitter is hence to introduce conditions on the computed schedule preventing flows from interleaving, thus guaranteeing the isolation of flows (or frames) in the transmitting queues. In other words, the constraints must either enforce a deterministic order of the flows in queues or place flows in different queues such that each frame is dispatched deterministically according to the schedule of the associated timed gate. In this case, the scheduler either enforces two flows arriving during interfering intervals to be placed in different queues (Figure 2(b)), or else allow them to be placed in the same queue but ensure that the intended order and transmission time for all frames of the flows are preserved (Figure 2(c)).

We first assume that the two considered flows arriving at the same device will be queued in the same egress queue and later relax this assumption and formulate the complete 802.1Qbv isolation constraint for high-criticality flows enabling deterministic behaviour in the time domain.

Let $s_i^{[v_a,v_b]}$ and $s_j^{[v_a,v_b]}$ be, respectively, the flow instances of $s_i \in \mathcal{S}$ and $s_j \in \mathcal{S}$ scheduled on the outgoing link (and hence egress port) $[v_a, v_b]$ of device v_a. We remind the reader the equivalence between link $[v_a, v_b]$ and the respective egress port and its associated output queues. Flow s_i arrives at the device v_a from some device v_x on link $[v_x, v_a]$. Similarly, flow s_j arrives from another device v_y on incoming link $[v_y, v_a]$. If s_i and s_j would arrive from the same device, constraint (1) would ensure that the frames of the flows will not overlap in the time domain.

Ideal scenario. Let's assume a perfect network where no frame is lost or discarded and all flows are sent constantly in full size by their sending nodes (i.e. constant payload size). In order to guarantee the desired output schedule, it would suffice to ensure that any two individual frames of different flows are not scheduled to arrive at the same time, hence having a deterministic queuing order. The isolation condition in an ideal network is

$$
\begin{aligned}
&\forall [v_a, v_b] \in \mathcal{L}, \forall s_i^{[v_a,v_b]}, s_j^{[v_a,v_b]} \in \mathcal{S}, i \neq j,\\
&\forall f_{i,k}^{[v_a,v_b]} \in \mathcal{F}_i^{[v_a,v_b]}, \forall f_{j,l}^{[v_a,v_b]} \in \mathcal{F}_j^{[v_a,v_b]},\\
&\forall \alpha \in [0, hp_i^j/s_i.T - 1], \forall \beta \in [0, hp_i^j/s_j.T - 1] :\\
&\Big(f_{i,k}^{[v_x,v_a]}.\phi \times [v_x, v_a].mt + \alpha \times s_i.T + [v_x, v_a].d + \delta \leq\\
&f_{j,l}^{[v_y,v_a]}.\phi \times [v_y, v_a].mt + \beta \times s_j.T + [v_y, v_a].d\Big) \vee\\
&\Big(f_{j,l}^{[v_y,v_a]}.\phi \times [v_y, v_a].mt + \beta \times s_j.T + [v_y, v_a].d + \delta \leq\\
&f_{i,k}^{[v_x,v_a]}.\phi \times [v_x, v_a].mt + \alpha \times s_i.T + [v_x, v_a].d\Big),
\end{aligned}
$$

where $hp_i^j \stackrel{def}{=} lcm(s_i.T, s_j.T)$ is, as before, the hyperperiod of flows s_i and s_j. The condition imposes that individual frames of any two flows sharing a queue on device $[v_a, v_b]$ are sent from their sending devices ($[v_x, v_a]$ and $[v_y, v_a]$) in such a way that they would arrive at the respective ports of the receiving device $[v_a, v_b]$ in a deterministic order.

Flow Isolation Constraint. In reality, frame loss can occur, periodic payload size may vary over time, or frames might be dropped due to e.g. time-based ingress policing (e.g. IEEE 802.1Qci). In these cases there may be instances where a deterministic behaviour on the egress port cannot be guaranteed. Consider the case where two frames from two different flows are scheduled to arrive one after the other at the switch. Both are placed in the desired order in the same queue. At a later point in time, after both have been placed in the queue, the timed gate of the queue is opened for the first frame (belonging to the first flow) and only at a later point in time for the second frame. If the first frame is lost, the frame from the second flow will take its place in the queue and be transmitted in the time slot originally reserved for the first flow leading to non-determinism and jitter. To guarantee isolation in a real network, we first consider isolating flows in the time domain assuming that they share the same egress queue. Hence, we formulate constraint (4) to enforce a correct ordering of flows, i.e., if a frame of a given flow has entered a queue, no frame of another flow may arrive at the queue until all frames of the previous flow have been fully dispatched. We remind the reader that the first and last frame of a flow, already ordered by their transmission time (i.e. scheduled offset), are defined by $f_{i,1}^{[v_a,v_b]}$ and $last(\mathcal{F}_i^{[v_a,v_b]})$, respectively. Hence, we have

$$
\begin{aligned}
&\forall [v_a, v_b] \in \mathcal{L}, \forall s_i^{[v_a,v_b]}, s_j^{[v_a,v_b]} \in \mathcal{S}, i \neq j,\\
&\forall \alpha \in [0, hp_i^j/s_i.T - 1], \forall \beta \in [0, hp_i^j/s_j.T - 1] :\\
&\Big(last(\mathcal{F}_i^{[v_a,v_b]}).\phi \times [v_a, v_b].mt + \alpha \times s_i.T + \delta \leq\\
&f_{j,1}^{[v_y,v_a]}.\phi \times [v_y, v_a].mt + \beta \times s_j.T + [v_y, v_a].d\Big) \vee \qquad (4)\\
&\Big(last(\mathcal{F}_j^{[v_a,v_b]}).\phi \times [v_a, v_b].mt + \alpha \times s_j.T + \delta \leq\\
&f_{i,1}^{[v_x,v_a]}.\phi \times [v_x, v_a].mt + \beta \times s_i.T + [v_x, v_a].d\Big).
\end{aligned}
$$

The constraint ensures that once a flow has arrived at the receiving device $[v_a, v_b]$, no other flow can arrive at the same egress port until the first flow has been completely sent on the output port. Hence, individual frames of the first arriving flow will not be interleaved with any frames of another flow until the queue has been completely emptied.

Frame Isolation Constraint. Despite being generally faster to solve since not all interleavings of individual frames have to be considered, the flow isolation constraint is restrictive and may decrease the search space for valid schedules. This can be easily proven by considering an input where a high-rate flow has a period which is equal to or less than the combined transmission duration of all frames of a low-rate flow. In this case, there is at least one period instance of the high rate flow in which its frames have to interleave with the frames of the low-rate flow. Given the previous isolation constraint, this would not be schedulable if there is only one queue per port, i.e., the flows must be placed in different queues. The counterexample can be generalized for an arbitrary number of queues.

In order to avoid this, we relax the previous constraint to allow frames interleaving between flows in a queue while in the same time guaranteeing that the order on the output port is deterministic. We do this by ensuring that there are only frames of one flow in the queue at a time, i.e., frames from another flow may only enter the queue if the already

queued frames of the initial flow have been serviced. We enforce that, if two frames are from different flows, one can only be scheduled to arrive at the shared queue if the other has already been dispatched from that queue. The frame isolation condition for schedulability is then

$$\begin{aligned}
&\forall [v_a, v_b] \in \mathcal{L}, \forall s_i^{[v_a,v_b]}, s_j^{[v_a,v_b]} \in \mathcal{S}, i \neq j, \\
&\forall f_{i,k}^{[v_a,v_b]} \in \mathcal{F}_i^{[v_a,v_b]}, \forall f_{j,l}^{[v_a,v_b]} \in \mathcal{F}_j^{[v_a,v_b]}, \\
&\forall \alpha \in [0, hp_i^j/s_i.T - 1], \forall \beta \in [0, hp_i^j/s_j.T - 1] : \\
&\Big(f_{j,l}^{[v_a,v_b]}.\phi \times [v_a, v_b].mt + \alpha \times s_j.T + \delta \leq \\
&f_{i,k}^{[v_x,v_a]}.\phi \times [v_x, v_a].mt + \beta \times s_i.T + [v_x, v_a].d\Big) \vee \\
&\Big(f_{i,k}^{[v_a,v_b]}.\phi \times [v_a, v_b].mt + \beta \times s_i.T + \delta \leq \\
&f_{j,l}^{[v_y,v_a]}.\phi \times [v_y, v_a].mt + \alpha \times s_j.T + [v_y, v_a].d\Big),
\end{aligned} \tag{5}$$

For further use, we denote constraint (5) (or, if the flow isolation method is used, constraint (4)) between two flows as $\Phi(s_i^{[v_a,v_b]}, s_j^{[v_a,v_b]})$.

The above constraints ((4) and (5)) apply to frames in the same queue. However, as mentioned before, the scheduler may choose to place flows in different queues. Hence, the complete constraint for minimum jitter scheduling of high-criticality flows is

$$\begin{aligned}
&\forall [v_a, v_b] \in \mathcal{L}, \forall s_i^{[v_a,v_b]}, s_j^{[v_a,v_b]} \in \mathcal{S}, \\
&\Big(\Phi(s_i^{[v_a,v_b]}, s_j^{[v_a,v_b]})\Big) \vee \Big(s_i^{[v_a,v_b]}.p \neq s_j^{[v_a,v_b]}.p\Big),
\end{aligned} \tag{6}$$

with $s_i^{[v_a,v_b]}.p \leq [v_a, v_b].c$ and $s_j^{[v_a,v_b]}.p \leq [v_a, v_b].c$.

5. 802.1QBV CONFIGURATIONS

The constraints presented above apply to the m scheduled queues of a generic configuration for critical traffic in fully scheduled 802.1Qbv networks, namely $\{\mathcal{V}_{e+s}, \langle n, m, n-m \rangle\}$. We now briefly discuss the implications of a number of special configurations which include non-scheduled traffic and different configurations of the scheduled and priority queues.

Configuration 1: $\{\mathcal{V}_{e+s}, \langle 1, 1, 0 \rangle\}$ This case corresponds to devices in a network having one single queue per egress port operated as a scheduled queue. Therefore, condition $s_i^{[v_a,v_b]}.p \neq s_j^{[v_a,v_b]}.p$ is always false and, hence, high-criticality flows are scheduled to be completely sequential in the time domain. This case amounts to serializing incoming traffic that converges on the same egress port reproducing a similar behaviour to bus systems like, e.g. TTP [23].

Configuration 2: $\{\mathcal{V}_{e+s}, \langle n, 1, n-1 \rangle\}$ This is a generalization of case 1, in which high-criticality flows are assigned to the scheduled queue while non-scheduled flows use one of the $n-1$ available priority queues. The timely behaviour of non-scheduled flows depends on queue priorities and arrival patterns [12, 14]. This configuration corresponds to the special case presented in [31] compatible with the urgency-based scheduler approach proposed by the authors. This configuration is suitable for legacy AVB systems that need to be augmented with a few additional high-criticality flows requiring tighter temporal bounds.

Configuration 3: $\{\mathcal{V}_{e+s}, \langle n, n, 0 \rangle\}$ Contrary to case 2, all of the queues are designated as scheduled queues but we assign $n-1$ queues for scheduled traffic, increasing the solution space for critical traffic. The remaining scheduled queue (BE-queue) is reserved for non-scheduled flows, allowing two alternative approaches. On one hand, leaving its timed gate permanently open may result in interference to scheduled traffic due to occasional shuffling caused by ongoing transmissions originated just before the opening of scheduled queues. This can be included in the schedulability conditions by extending the duration of individual scheduled frames assuming that, in the worst case, one non-scheduled frame of MTU size can shuffle each critical frame. On the other hand, we propose to generate a schedule for the timed gate of the BE-queue by negating the combined schedule of all scheduled queues, i.e., for each gate open event of any other queue of the port, we create a gate close event for the BE-queue, and vice-versa. If need be, depending of device implementation, gate closing events for the BE-queues may be decreased by the duration of MTU-sized frame in order to guarantee that no unscheduled frame may interfere scheduled frames. In Figure 3 we depict a simple example where the BE-queue schedule is generated from the combined schedules of all other queues but with the duration of an MTU BE frame (denoted by ρ in the figure) subtracted from the gate open events of the combined schedule.

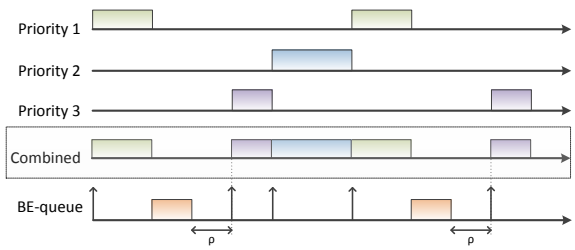


Figure 3: *Example of generating BE-queue schedules.*

Note that, if all egress ports would have as many queues as there are incoming scheduled flows, each could be assigned to its own dedicated queue. Hence, the problem is similar to scheduling TTEthernet, since $s_i^{[v_a,v_b]}.p \neq s_j^{[v_a,v_b]}.p$ is always true. In this case, each queue will behave as a "large" buffer, sufficient to accommodate all frames of each flow instance, guaranteeing isolation between flows through the schedule.

Configuration 4: $\{\mathcal{V}_{e+s}, \langle n, m, n-m \rangle\}$ This scenario is suitable for high-criticality applications that feature both scheduled and non-scheduled traffic. The separation in scheduled and non-scheduled queues influences both the solution space for high-criticality flows and worst-case latency of non-scheduled flows for analysis methods like network calculus [12, 14] or the trajectory approach [4]. If $m = 1$, we have configuration 2 in which non-scheduled traffic can be optimized at the expense of solution space for high-criticality traffic. On the other hand, if $m = n-1$, the solution space for high-criticality traffic is maximized but the pessimism of the latency bounds of non-scheduled flows may be higher. In Section 6.1 we discuss optimization criteria for the general case of this configuration.

Configuration 5: $\{\mathcal{V}_{e+s}, \langle n, 0, n \rangle\}$ This case mimics the behaviour of a standard AVB (IEEE 802.1BA) network in which flows are serviced according to the priority assigned to their traffic class, which is equivalent to setting the timed-gates to be permanently opened. End-to-end properties of the non-scheduled flows are subject to complementary analysis methods like e.g. [6, 9, 12].

6. 802.1QBV SCHEDULER

Satisfiability Modulo Theories (SMT) are designed to determine the satisfiability of first-order logical formulas against certain background theories like linear integer arithmetic ($\mathcal{LA}(\mathbb{Z})$) or bit-vectors ($\mathcal{BV}$) [3, 29]. On top of checking satisfiability, SMT solvers also provide a *model* for the given satisfiable context which represents one solution (out of a set of potentially multiple feasible solutions) for the given variables and constraints. NP-complete scheduling problems that exhibit combinatorial characteristic and have arithmetic constraints, like the one addressed in this paper, present a suitable use-case for constraint-satisfaction SMT solving in linear arithmetic [1, 11]. Using SMT for solving deterministic network scheduling problems was first proposed by Steiner in [33].

The aim of our scheduling algorithm for 802.1Qbv is to find values for all individual frame offsets and queue assignments in each respective egress port of flows routed along the network such that the set of constraints are met. We define both frame offsets and queue assignment indexes as integer variables to the SMT context and generate assertions (in linear arithmetic) that correspond to the constraints defined in the previous sections. These frame offsets represent the open (and close events) for the timed-gate of the respective queue assignment. The transformation from frame offset and queue index into gate open close events is straightforward, i.e., the offset represents the gate open event for the given queue index and the gate close event is marked by the duration of the respective frame.

When it comes to scheduling problems, the scalability of the SMT approach depends on several key factors, analyzed in detail in [8]. However, an additional consideration, specific to 802.1Qbv, is the interleaving of different flows in a given queue. To improve scalability, especially when scheduling large networks, Steiner introduced an incremental backtracking algorithm in [33] which we also use here in a modified form. Our approach attempts to schedule one flow at a time by adding the flow variables and constraints to the SMT context and trying to solve the problem with the added constraints. If a solution is found, the schedule for the flow is fixed by asserting the constant value provided by the SMT model into the context. This repeats until either the complete schedule is found (i.e. all flows are scheduled) or an incremental step is deemed unfeasible by the solver. In the latter case, the constraints of the current flow could not be satisfied with the previous context, so the SMT context is backtracked by removing the last scheduled flow and reintroducing it together with the unfeasible step in question. Backtracking repeats as long as the merged step results in an unfeasible problem. In the worst case, the algorithm schedules all flows in one single step. However, in the average case, there is a significant performance improvement that can be substantial especially when network utilization is low. In the case of infeasiblity, we save the state of the context with the highest number of solved flows. Hence, even if the whole set of flows was not schedulable, we still provide a partial solution for a subset of the given flows.

6.1 Optimization

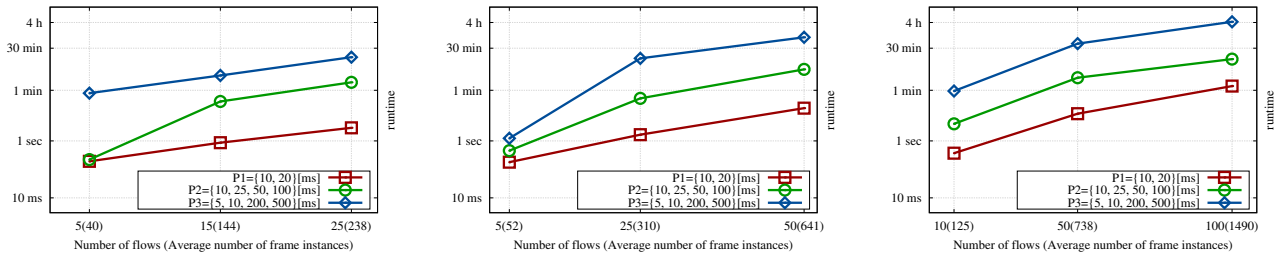
SMT solvers return the first encountered valid solution, if one exists, for the given context out of the set of (potentially) multiple solutions. Scheduling solutions, like the one presented here, would benefit even more if at the same time the provided solution is optimized with respect to certain properties of the system. Optimization solvers, like Gurobi [16] or GLPK [15], are explicitly designed to solve constrained optimization problems and can be used to tackle optimized scheduling (cf. [8] for an example). In recent years, a new field, called Optimization Modulo Theories (OMT), has emerged where certain SMT solvers are augmented with optimization capabilities [5, 24, 30]. A typical optimization objective used in industry is minimizing end-to-end latency of selected flows. The minimization objective is easily expressed based on constraint (3) from Section 4.2 and has been discussed in previous work (cf. [8]). Here, we focus on optimization opportunities arising from the specifics of IEEE 802.1Qbv.

In the presented method we have seen how scheduled flows are isolated either in the time domain or in different queues in order to avoid interleaving. Scheduling along the time domain is constrained by the end-to-end latency requirements and periods of the set of flows, while the selection of queues is limited by the device capacity and the necessity to integrate other traffic classes. We have also seen that non-scheduled flows, may benefit from a larger number of priority queues in the post-analysis step. On the other hand, a small number of scheduled queues available to scheduled flows, reduces the solution space due to the rather stringent isolation constraints (cf. constraints (4) and (5)). Hence, any pre-assignment of queues to traffic classes introduces a trade-off between schedulability of high-criticality flows and timeliness properties and flexibility for non-scheduled traffic.

The inclusion of optimization, however, opens up alternative design space exploration mechanisms. Therefore, we propose an optimization objective to find the minimal number of queues required for scheduled traffic such that a valid solution is still feasible (i.e. schedulability is guaranteed). This number represents a design input for the assignment of queues to traffic classes on a per-device basis. The optimization condition is easy to express using our approach by having an additional variable ($[v_a, v_b].\kappa$) for each device representing the number of required scheduled queues for high-criticality flows for each egress port. Minimizing the number of used queues can be specified in different ways depending on the OMT implementation and design goals. We define a global objective to minimize the accrued sum of the number of queues used per egress port, i.e., minimize $\sum_{[v_a,v_b]\in\mathcal{L}}[v_a, v_b].\kappa$. Complementary, OMT solvers like Z3 (v4.4.1) offer the possibility to incorporate multiple different objectives either optimized independently or combined using Pareto fronts or lexicographic priorities [5]. Thus, additionally to the constraints defined in Section 4.1 and 802.1Qbv constraint (6), we add, $\forall[v_a, v_b] \in \mathcal{L}$

$$\begin{aligned} &\text{minimize} && [v_a, v_b].\kappa \\ &\text{subject to} && [v_a, v_b].\kappa \leq [v_a, v_b].c, \\ & && s_i^{[v_a,v_b]}.p \leq [v_a, v_b].\kappa, \forall s_i^{[v_a,v_b]} \in \mathcal{S}. \end{aligned}$$

In this case the solution given by the solver will not only return a valid schedule (in terms of frame offsets and queue assignments per device) but also the minimum number of queues per device/port which are required to fulfil the computed schedule for high-criticality flows. The remaining queues are then guaranteed to optimize the trade-off for non-scheduled flows. As can be seen, there is no assumption on the value of $[v_a, v_b].c$ which represents the maximum



(a) Simple topology (3 ES, 1 SW) (b) Medium topology (5 ES, 2 SW) (c) Complex topology (7 ES, 5 SW)

Figure 4: *Runtime with varying number of flows and periods and different network topologies.*

number of queues. We can therefore also reformulate the optimization problem in terms of design space exploration, i.e., given an unschedulable system, what is the minimum number of queues in each device/port that are necessary in order for the problem to become schedulable.

7. EXPERIMENTS

For the experimental evaluation we use the Z3 v4.4.1 solver (64bit) [10] which features algorithms for solving linear optimization objectives in addition to the normal SMT functionality [5]. We reproduced additionally several tests without optimization with Yices v2.4.2 (64bit) [13] using quantifier-free linear integer arithmetic ($\mathcal{LA}(\mathbb{Z})$) as the background theory. Since we observe similar trends as in the Z3 experiments and a comparison of SMT solvers is outside the scope of this paper, we omit the evaluation with Yices and concentrate on Z3. All experiments were run on a 64bit 4-core[3] 3.40GHz Intel *Core-i7* PC with 4GB memory.

We analyze the scalability and schedulability aspects of our methods over a number of synthetic configurations on top of three predefined network topologies ranging from 3 end-systems connected to one switch to 7 end-systems connected through 5 switches (via 1Gbit/s links with a $1\mu sec$ macrotick granularity). We set the time-out value for a run to 5 hours. As shown in [8], a higher utilization of the network links results in a more difficult problem set for the SMT solver. Hence, we keep the size of the topologies relative small in comparison to the number of flows (and frames) in order to achieve a higher utilization on the links.

In the first set of experiments (Figure 4) we show the scalability of the frame isolation method (using the incremental backtracking implementation with step size of 1 flow) when varying the problem set in 3 dimensions, namely in topology size, number of flows and flow periods (chosen randomly from 3 sets of predefined periods). The configuration we use for the scalability tests is $\{\mathcal{V}_{e+s}, \langle 8, 8, 0\rangle\}$, the data size for each flow is chosen uniformly between 2 and 8 MTU-sized frames, and the senders and receivers of the flows are also chosen randomly from the set of end-systems in the network.

The logarithmic y-axis shows the runtime and the x-axis the number of flows and average number of frames scheduled. We observe that the period set has a significant impact on the scalability, i.e. a more complex relationship between the periods of the flows, and hence, a higher hyperperiod, results

[3]Note that the current implementation of Z3 (and Yices) does not take advantage of the multi-core nature of the CPU as it runs in a single core.

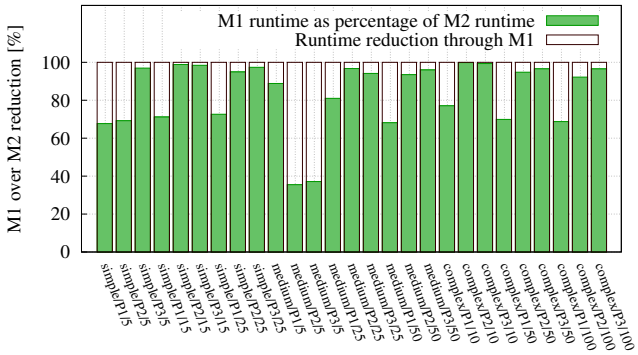


Figure 5: *Flow vs. frame isolation runtime comparison.*

in significantly higher runtime of the algorithm. As is to be expected with NP-complete problems, the runtime increases exponentially with an increase in the number of flows and frames that have to be scheduled. For e.g., using the complex topology (Figure 4(c)), we schedule 10 flows with 118 frame instances and period configuration $P1$ in under 1 sec while scheduling 100 flows with 1394 frame instances and with period configuration $P3$ takes over 4 hours to solve.

Figure 5 shows the reduction of the runtime when using the flow isolation approach as a percentage of the runtime of the frame isolation method for each of the above inputs. Using the flow isolation method, we expect on average a reduced runtime of the scheduler at the expense of schedulability. We replicating the above experiments using the flow isolation (constraint (4)) method, denoted by $M1$ in Figure 5, instead of the frame isolation constraint (constraint (5)), denoted by $M2$. As can be seen, in these experiments the flow isolation method is always faster than the frame isolation approach. In order to confirm the trend, we have run additional experiments with 381 randomly generated test cases varying between the 3 topologies and period sets presented above and with up to 1000 flows and data sizes from 500 up to 8000 bytes. Out of these, 17 reached the time-out of 5 hours with either $M2$[4] or with both methods and were deemed infeasible. For the remaining 336 inputs, $M1$ was on average 13% faster than $M2$ with a median of 8.03%. The accumulated runtime of all successful runs was around 36.7 hours for $M1$ and just over 59 hours for $M2$, which amounts to a 30.73% improvement.

[4]In some isolated cases, the frame isolation method timed out while the flow isolation found a solution within minutes.

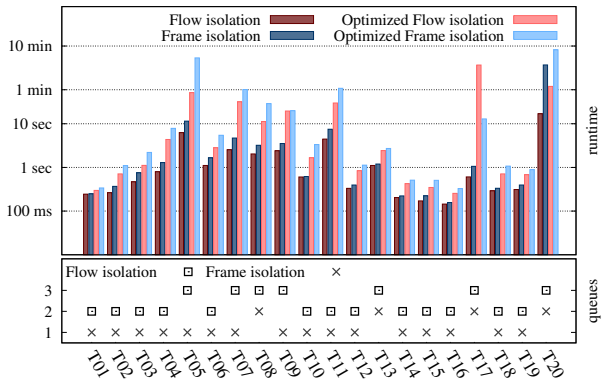


Figure 6: *Flow vs. frame isolation w/ and w/o optimization.*

Although in some cases the flow isolation method reduces the execution time significantly and may generally be preferable to the frame isolation approach (e.g. if schedulability is not likely to be an issue), it brings mostly moderate (linear) improvements in runtime, especially in relation to the exponential nature of the problem.

For the schedulability tests we generate random test cases with either simple, medium or complex topologies and with 3, 4 or 5 flows. The periods of the flows are chosen randomly between two different orders of magnitude, i.e., one set of periods is $\{1ms, 2ms, 3ms, 5ms, 10ms\}$, while the others have values from the set $\{100\mu sec, 150\mu sec, 200\mu sec, 500\mu sec\}$. We scale the number of frames per flow accordingly, i.e. flows with periods over $1ms$ have 10, 20, 40, 60 or 100 frames per stream and high-rate flows have either 1 or 2 frames. Through this, we try to generate the scenario described in Section 4.2 in which frames of different flows have to interleave if scheduled in the same egress queue. In Figure 6 we show the runtime of the scheduler with and without optimization objectives using both flow and frame isolation methods. As optimization objective we minimize the accrued sum of the number of queues used per egress port, therefore, aiming at the lowest overall number of required queues per egress port. From all randomly generated test cases we selected 20 for which the minimum number of queues required was different between the flow and frame isolation methods. In order to obtain a global minimum, we schedule all flows at once (i.e. without incremental steps) when using optimization. For the non-optimized runs we use the incremental version of our algorithm.

As expected, the runtime is generally higher when optimizing regardless of whether we use flow or frame isolation without optimization. In most cases, we observe that the frame isolation method requires one single queue per egress port for any device to satisfy the schedule while the flow isolation method usually needs more than one queue in several ports/devices. This also means that configurations 1 or 2 (defined in Section 5) would not render a feasible solution using the flow isolation method due to the limited amount of queues for high-criticality traffic flows. For some use-cases ($T05$, $T07$, and $T09$), the flow isolation method requires 3 queues for some devices, while the frame isolation remains at 1 queue per port for all devices. For other cases ($T08$, $T13$, $T17$ and $T20$) the frame isolation method requires a minimum of 2 queues for some ports (still less than the flow isolation for the respective use-cases).

8. RELATED WORK

Scheduling critical traffic in deterministic networks using SMT solvers was first proposed by Steiner in [33] and extended in our previous work [7, 8] to include scheduling of preemptive tasks running on the end-system nodes. Scheduling traffic with different criticality classes in TTEthernet while optimizing the end-to-end delay of rate-constrained flows has been studied in [36].

Scheduling problems, with and without optimization, for other proprietary technologies like PROFINET, FlexRay, and TTP have been address in [17, 18, 26, 37]. At the other end of the spectrum, methods for determining worst-case end-to-end latencies for non-scheduled networks (like AFDX or AVB) have been developed for example in [4, 12, 14]. More recent approaches [9, 28] compute AVB worst case traversal times using network calculus. A new asynchronous traffic class is introduced in [31] within the context of TSN where the goal is to offer compositional analysis capabilities by maintaining the traffic pattern between egress ports along the route. This approach considers standard AVB priority behaviour since it assumes that the timed gates of the egress port queues are open at all times.

Meyer et al. [25] study the interference of time-triggered communication on AVB traffic in TSN networks when synchronous traffic is added to the credit-based shaper defined in 802.1Qbv. The simulation framework presented in the paper does not address the scheduling problem but assumes that an already existing TT schedule is created under the condition that TT frames circumvent the queues of the output port and are selected by the credit-based shaper with a higher priority than AVB streams. This approach assumes non-standard-compliant 802.1Qbv capabilities of devices combining a TTEthernet-like buffered approach with the scheduled queue architecture of TSN.

Alderisi et al. [2] introduce a new type of traffic class with real-time guarantees called Scheduled Traffic (ST) which corresponds to our high-criticality class. Similar to [25], the paper considers a strict isolation of the mechanisms handling ST and AVB traffic to ensure non-interference. As opposed to our approach, both papers view the scheduled or time-triggered traffic class and the AVB class as fundamentally dichotomic and hence the two classes need separate (and different) hardware mechanisms. Additionally, the authors do not address the underlying scheduling problem introduced by the timed gates of the priority queues.

To the best of our knowledge, this is the first study which discusses the functional parameters of 802.1Qbv devices impacting the capabilities to control the temporal behaviour of flows and the resulting scheduling problem concerning the timed gate functionality of egress ports for high-criticality traffic.

9. CONCLUSION

In this work we addressed the scheduling problem arising from the IEEE 802.1Qbv extension on multi-hop fully switched TSN networks, presenting several methods for computing static schedules via Satisfiability Modulo Theories (SMT). We identified key functional parameters affecting the behaviour of 802.1Qbv communication and, based on a generalized configuration of these parameters for real-time traffic, we derived constraints for creating correct offline schedules guaranteeing low and bounded jitter as well as deterministic end-to-end latencies for critical communication

flows. Furthermore, we discussed several optimization directions as well as concrete system configurations which open up a number of trade-offs both at runtime as well as in the design phase of the system. To demonstrate our approach, we have performed an evaluation in terms of scalability and schedulability via synthetic network workloads and system configurations.

References

[1] Abraham, E., and Kremer, G. Satisfiability checking: Theory and applications. In *Proc. SEFM* (2016), vol. 9763 of *LNCS*, Springer International Publishing.

[2] Alderisi, G., Patti, G., and Bello, L. L. Introducing support for scheduled traffic over IEEE audio video bridging networks. In *Proc. ETFA* (2013), IEEE Computer Society.

[3] Barrett, C., Sebastiani, R., Seshia, S., and Tinelli, C. Satisfiability modulo theories. In *Handbook of Satisfiability*, vol. 185. IOS Press, 2009.

[4] Bauer, H., Scharbarg, J., and Fraboul, C. Improving the worst-case delay analysis of an AFDX network using an optimized trajectory approach. *Industrial Informatics, IEEE Transactions on 6*, 4 (2010).

[5] Bjørner, N., Phan, A., and Fleckenstein, L. νz - an optimizing SMT solver. In *Proc. TACAS* (2015), Springer.

[6] Bordoloi, U. D., Aminifar, A., Eles, P., and Peng, Z. Schedulability analysis of ethernet avb switches. In *Proc. RTCSA* (2014), IEEE Computer Society.

[7] Craciunas, S. S., and Serna Oliver, R. SMT-based task- and network-level static schedule generation for time-triggered networked systems. In *Proc. RTNS* (2014), ACM.

[8] Craciunas, S. S., and Serna Oliver, R. Combined task- and network-level scheduling for distributed time-triggered systems. *Real-Time Systems 52*, 2 (2016), 161–200.

[9] De Azua, J. A. R., and Boyer, M. Complete modelling of AVB in network calculus framework. In *Proc. RTNS* (2014), ACM.

[10] De Moura, L., and Bjørner, N. Z3: An efficient SMT solver. In *Proc. TACAS* (2008), Springer-Verlag.

[11] De Moura, L., and Bjørner, N. Satisfiability modulo theories: Introduction and applications. *Commun. ACM 54*, 9 (2011), 69–77.

[12] Diemer, J., Thiele, D., and Ernst, R. Formal worst-case timing analysis of ethernet topologies with strict-priority and AVB switching. In *Proc. SIES* (2012), IEEE Computer Society.

[13] Dutertre, B. Yices 2.2. In *Proc. CAV* (2014), vol. 8559 of *Lecture Notes in Computer Science*, Springer, pp. 737–744.

[14] Frances, F., Fraboul, C., and Grieu, J. Using network calculus to optimize the AFDX network. In *Proc. ERTS* (2006).

[15] GLPK. GNU Linear Programming Kit. http://www.gnu.org/software/glpk/. retrieved 20-Jul-2016.

[16] Gurobi Optimization, I. Gurobi optimizer reference manual, version 6.0, 2014. retrieved 12-Jan-2015.

[17] Hanzalek, Z., Burget, P., and Šucha, P. Profinet IO IRT message scheduling. In *Proc. ECRTS* (2009), IEEE.

[18] Huang, J., Blech, J. O., Raabe, A., Buckl, C., and Knoll, A. Static scheduling of a time-triggered network-on-chip based on SMT solving. In *Proc. DATE* (2012), IEEE.

[19] Institute of Electrical and Electronics Engineers, Inc. 802.1Qbv - Enhancements for Scheduled Traffic. http://www.ieee802.org/1/pages/802.1bv.html, 2016. Draft 3.1.

[20] Institute of Electrical and Electronics Engineers, Inc. Time-Sensitive Networking Task Group. http://www.ieee802.org/1/pages/tsn.html, 2016. retrieved 06-Jul-2016.

[21] Issuing Committee: As-2d2 Deterministic Ethernet And Unified Networking. SAE AS6802 Time-Triggered Ethernet. http://standards.sae.org/as6802/, 2011. retrieved 20-May-2014.

[22] Kopetz, H., and Bauer, G. The time-triggered architecture. *Proceedings of the IEEE 91*, 1 (2003), 112–126.

[23] Kopetz, H., and Grunsteidl, G. TTP - a time-triggered protocol for fault-tolerant real-time systems. In *Proc. 23rd IEEE International Symposium on Fault-Tolerant Computing (FTCS-23)* (June 1993), pp. 524–533.

[24] Li, Y., Albarghouthi, A., Kincaid, Z., Gurfinkel, A., and Chechik, M. Symbolic optimization with SMT solvers. *SIGPLAN Not. 49*, 1 (Jan. 2014).

[25] Meyer, P., Steinbach, T., Korf, F., and Schmidt, T. Extending IEEE 802.1 AVB with time-triggered scheduling: A simulation study of the coexistence of synchronous and asynchronous traffic. In *Proc. VNC* (2013), IEEE Computer Society.

[26] Pop, P., Eles, P., and Peng, Z. Schedulability-driven communication synthesis for time triggered embedded systems. *Real-Time Syst. 26*, 3 (2004), 297–325.

[27] Prytz, G. A performance analysis of EtherCAT and PROFINET IRT. In *Proc. ETFA* (2008), IEEE Computer Society.

[28] Queck, R. Analysis of ethernet AVB for automotive networks using network calculus. In *Proc. ICVES* (2012), IEEE Computer Society.

[29] Sebastiani, R. Lazy satisfiability modulo theories. *JSAT 3*, 3-4 (2007), 141–224.

[30] Sebastiani, R., and Trentin, P. OptiMathSAT: A Tool for Optimization Modulo Theories. In *Proc. CAV* (2015), vol. 9206 of *LNCS*, Springer.

[31] Specht, J., and Samii, S. Urgency-based scheduler for time-sensitive switched ethernet networks. In *Proc. ECRTS* (2016), IEEE Computer Society.

[32] Steinbach, T., Lim, H.-T., Korf, F., Schmidt, T., Herrscher, D., and Wolisz, A. Tomorrow's in-car interconnect? a competitive evaluation of IEEE 802.1 AVB and Time-Triggered Ethernet (AS6802). In *Proc. VTC* (2012), IEEE Computer Society.

[33] Steiner, W. An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks. In *Proc. RTSS* (2010), IEEE Computer Society.

[34] Steiner, W., Bauer, G., Hall, B., and Paulitsch, M. TTEthernet: Time-Triggered Ethernet. In *Time-Triggered Communication*, R. Obermaisser, Ed. CRC Press, Aug 2011.

[35] Steiner, W., and Dutertre, B. The TTEthernet synchronisation protocols and their formal verification. *Int. J. Crit. Comput.-Based Syst. 4*, 3 (2013).

[36] Tamas-Selicean, D., Pop, P., and Steiner, W. Synthesis of communication schedules for TTEthernet-based mixed-criticality systems. In *Proc. CODES+ISSS* (2012), ACM.

[37] Zeng, H., Zheng, W., Di Natale, M., Ghosal, A., Giusto, P., and Sangiovanni-Vincentelli, A. Scheduling the flexray bus using optimization techniques. In *Proc. DAC* (2009), ACM.