An Overview of Scheduling Mechanisms for Time-sensitive Networks

Silviu S. Craciunas Ramon Serna Oliver
TTTech Computertechnik AG
Schönbrunner Straße 7
Vienna, Austria
Email: {scr, rse}@tttech.com

***Abstract*—Ethernet has now become the preferred communication mechanism for a wide range of application domains that do not impose strict real-time constraints. Within the real-time domain, safety-critical timing aspects have been introduced on top of Ethernet by means of technologies like TTEthernet and Time-Sensitive Networks (TSN). In such technologies communication is aligned to an offline and statically configured schedule (the synthesis of which is an NP-complete problem) guaranteeing contention-free frame transmissions along the network devices as well as real-time end-to-end behavior of individual frames or entire traffic classes.**

In this overview we discuss results on scheduling TSN- and TTEthernet-compliant multi-hop switched networks using Satisfiability Modulo Theories (SMT) and, alternatively, Optimization Modulo Theories (OMT) solvers. We identify and analyze key functional parameters affecting the deterministic behaviour of real-time communication under TSN and TTEthernet and derive the required constraints for computing offline schedules guaranteeing low and bounded jitter and deterministic end-to-end latency for critical communication streams.

I. Introduction

The need for end-to-end real-time guarantees in distributed systems has long been a requirement in the aerospace domain. In recent years other domains like industrial automation and automotive systems show an increasing demand for deterministic communication behavior. Deterministic networks have been traditionally build on top of technologies like TTEthernet (SAE AS6802 [1, 2]), PROFINET, and EtherCAT [3] among others. Soft-real time aspects have been standardized for IEEE 802 networks through the IEEE 802.1BA Audio/Video Bridging (AVB) standard. Driven by an increasing need for standardized mechanisms allowing deterministic behavior that goes beyond the Quality of Service guarantees provided by IEEE 802.1BA, the IEEE 802.1 Time Sensitive Networking (TSN) task group [4] is in the process of standardizing time-sensitive capabilities over IEEE 802 networks. The key drivers for real-time communication over standard Ethernet include a fault tolerant clock synchronization mechanism (IEEE 802.1ASrev [4]) that provides a network-wide clock reference and mechanisms enabling a global communication schedule governing the end-to-end timing of communication streams (IEEE 802.1Qbv [5]). A stream is a communication from a sender (talker) to one or more receivers (listeners) with or without end-to-end timing requirements. In case of critical streams, the communication is defined by a payload size and a period in addition to the sender and receivers devices.

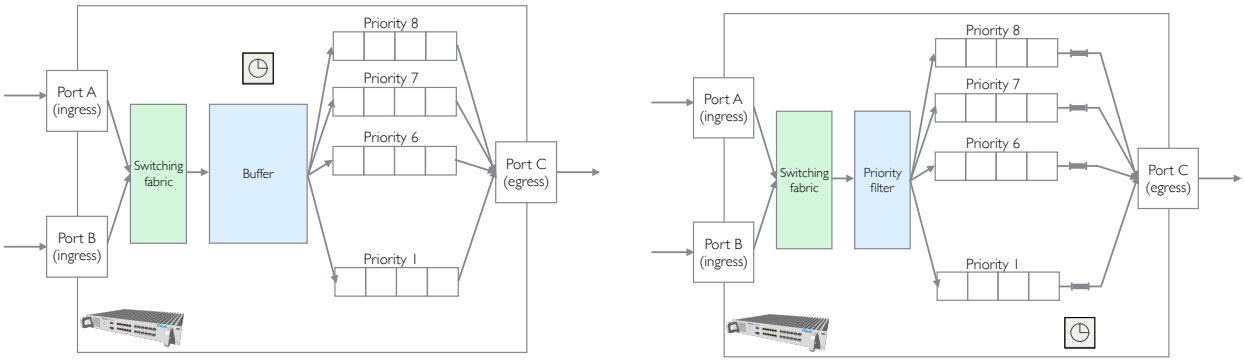
In this overview based on our previous work [6, 7] we describe the scheduling principles governing both TTEthernet and TSN networks that result in a deterministic behavior for critical streams. In both cases we assume that a synchronization mechanism is in place that provides a global clock reference with a known and bounded maximal deviation (the so-called *precision*). We start by describing deterministic networks (Section II) and the formalization of the traffic and network model (Section III). We then present our results from previous work regarding global and local constraints that enable deterministic scheduling with bounded latency and jitter in Section IV. Using these constraints, we present SMT-based mechanisms to find correct network schedules together with possible optimization criteria and conclude the overview in Section VI.

II. Deterministic Networks

Deterministic networks refer to networks where at least a subset of the communication has to adhere to some type of hard real-time behavior. By hard real-time in this context we mean that the temporal behavior of communication is deterministic over the lifetime of the system. In other words, we want to ensure that safety-critical traffic behaves deterministically irrespective of other loads in the network. In this section we briefly describe the mechanisms that enable this deterministic behavior in both TTEthernet and TSN networks (c.f. Figure 1), focusing on the fundamental differences between the two.

A. TTEthernet

TTEthernet is an extension to standard Ethernet currently used in mixed-criticality real-time applications. Each device has the capability to dispatch frames according to a local schedule derived from a global communication scheme, the *tt-network-schedule*. The schedule defines transmission and reception time windows for each time-triggered frame being transmitted between nodes. The tt-network-schedule is typically built offline, accounting for the maximum end-to-end latency, message length, as well as constraints derived from resources and physical limitations, e.g., maximum frame buffer capacity. Inside the networks devices, a mechanism



(a) TTEthernet switch.

(b) TSN switch.

Figure 1: *A simplified representation of TTEthernet and TSN switches*

is in place that ensures that frames are transmitted from an internal buffer at the time specified by the schedule (c.f. Figure 1(a)). If a device in the network does not support this functionality, but is used to route critical streams, the real-time guarantees cannot be enforced. The critical traffic is isolated from non-critical traffic (rate-constrained or best-effort) in different queues, where the critical traffic always has a static assignment to a high-priority queue that guarantees a bounded delay in each device. Note that we say bounded delay because there can be in the worst case some delay due to low priority traffic (at most one frame) if the lower-priority frame starts being transmitted just before the scheduling point of a critical frame. Since there is no preemption mechanism in place, the low priority frame cannot be interrupted and hence can delay a critical frame. The two methods put in place to mitigate for this delay are timely block and shuffling which have been implemented in practice [8, p. 42-5]. With shuffling the scheduling window of critical frames includes the maximum frame size of other frames that might interfere with the sending of these frames, while the timely block method will prevent any low-priority frame to be send if it would delay a scheduled critical frame [8, p. 42-5], [2], hence reducing the duration of the delay. Note that, due to the fault-tolerant clock synchronization protocol used (SAE AS6802 [1, 9]), there will always be a bounded delay that critical frames can experience due to the fact that synchronization frames have higher priority than critical frames.

The combination of these two elements allows for safety-critical traffic with guaranteed end-to-end latency and minimal jitter in co-existence with rate-constrained flows bounded to deterministic quality of service (QoS) and non-critical traffic (i.e. best-effort).

The important aspect to remember here is that TTEthernet devices dispatch frames according the the defined schedule, i.e. the TTEthernet schedule is defined on the level of individual frames.

B. Time-Sensitive Networks

Similar to TTEthernet, a requirement for deterministic behavior is that there is a network-wide synchronization protocol in place that provides a time reference for the runtime execution of the statically created schedule. TSN networks are envisioned to run based on the IEEE 802.1ASrev [4] time-synchronization protocol. One difference to AS6802 is that there is no requirement in the standard that the synchronization frames have to have the highest priority. In terms of scheduling, IEEE 802.1Qbv [5] defines a time-based shaper functionality enabling time-triggered communication [10] 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 (c.f. Figure 1(b)). Here it is important to note that the schedule is defined on the level of traffic classes and not of individual frames like in TTEthernet. Scheduling entire traffic classes as opposed to individual frames provides more flexibility for use-cases where strict timing constraints and determinism on the level of streams are not the most important aspects. Since the traffic class is defined in the PCP code of the VLAN tag of frames, using only IEEE 802.1Qbv cannot enable a fined-grained identification and control on the level of streams. Additional mechanisms like the per-stream identification and filtering (defined in IEEE 802.1Qci/802.1CB), allowing identification of frames based on a stream id and overriding of the traffic class encoded in the PCP code, are necessary if we want to achieve the same level of determinism as in TTEthernet. We will later see how this affects the scheduling constraints in comparison to TTEthernet.

III. Network and Traffic Model

We now formalize the network and traffic model based on [6, 7]. Both TTEthernet and TSN cases are multi-hop layer 2 switched Ethernet networks over *full-duplex* multi-speed physical links. The network model is defined similar to [6] and [11], 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 [6], a physical link $[v_a, v_b]$ is characterized by the tuple $\langle [v_a, v_b].s, [v_a, v_b].d, [v_a, v_b].mt \rangle$, 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. For TTEthernet, this definition is sufficient since the priority queues on the output ports of devices are used in a predefined manner, namely priority 7 will be reserved for critical traffic while all lower priorities are reserved for rate-constrained and best-effort traffic. The highest priority is reserved for network synchronization frames. For TSN networks we need to extend the definition with a new parameter, $[v_a, v_b].c$, representing the number of available scheduled queues in the device. The value for this parameter is set to at most 8 according to 802.1Qbv [5], however, there is no limitation on the number from the point of view of the scheduling constraints. The macrotick offers a trade-off between schedulability and runtime of the scheduling algorithm and specifies the granularity of the time-line used for scheduling on that specific link.

A stream (flow) is defined as a periodic data transmission from one sender (talker) to one or multiple receivers (listeners). In this overview we are only concerned with critical streams. We denote the set of critical streams by $\mathcal{S}$. Similar to [11], a stream $s_i \in \mathcal{S}$ from a sender node v_a to a receiver node v_b is expressed as $s_i = [[v_a, v_1], [v_1, v_2], \ldots, [v_{n-1}, v_n], [v_n, v_b]]$, where $v_1, v_2, \ldots, v_{n-1}, v_n$ represents the route of the stream through the network. Please note that the routing is a preliminary step done separately from scheduling. A stream is defined through 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 stream, respectively. An instance of a stream $s_i \in \mathcal{S}$ on an outgoing link $[v_a, v_b] \in \mathcal{L}$, i.e., sent from device v_a to device v_b, is denoted by $s_i^{[v_a,v_b]}$.

For TSN networks, we introduce an additional variable specifying the assigned queue for the instance of a stream on a particular device, i.e., $s_i^{[v_a,v_b]}.p$. Note that the queue variable is synonymous to the priority of the stream within the egress port of the respective device. We draw an equivalence between the egress port and its respective transmission link. TSN defines per-stream filtering and identification (IEEE 802.1Qci/802.1CB) allowing a stream instance to be assigned to different queues in different devices.

While in TTEthernet, the stream is limited to at most an MTU sized frame, in TSN, the data size of may exceed the Ethernet MTU size and be split into multiple frames of MTU size. To devise a model governing both TTEthernet and TSN, we define that each stream instance is associated with a set of frames, each with size less than or equal to the MTU size. For TTEthernet this set will have size 1. Maintaining our notation from [6, 7], we denote the set of frames $f_{i,j}^{[v_a,v_b]}$ of a stream 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 stream 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 [6, 7].

In [6] we introduced an additional variable to the frame definition, namely the period instance, in order to enable end-to-end latencies exceeding the period boundary. Although present in the implementation, we omit the period instance from the formalization in order to simplify the notation and refer the reader to [6] for details on how to include the variable in the constraints.

IV. Scheduling Constraints

Now that we have defined the traffic and network model for both TTEthernet and TSN we will proceed to describe based on [6, 7] the relevant constraints for creating correct schedules. The constraints are of 4 types. The basic Ethernet constraints arise from the inherent functionality of TTEthernet and TSN. For TSN there are special constraints that only define correctness for IEEE 802.1Qbv networks. Apart from these two types we have also user constraints that define specific behaviors like start and end times for streams and constraints that arise from particular implementations of TTEthernet or TSN devices. User constraints can be added on individual streams or sets of streams depending on system design or user requirements. We will just briefly mention some of them noting that the formalization is straightforward. *Start time constraints* can be applied to streams imposing that either the receiving or sending of a stream happens after a given start time. Conversely, *end time constraints* specify that either the receiving or sending of a stream happens before a given end time. These constraints are important for synchronizing the network schedule to the task schedule for producing and consuming applications if scheduling is done sequentially (e.g. [12, 13]). *Precedence constraints* between streams impose a sending and/or receiving order between different streams. A simplification of this user constraint is when all given streams shall have the same period. More complex precedence relations, which do not require the same rate on streams, have been studied in [14].

In this overview we will focus only on the first two types of constraints.

A. Basic Ethernet Constraints

The constraints can be thought of as a system of inequalities where the variables are the frame offsets (and queue assignment for TSN) that describe a correct temporal behavior for the communication streams. We adapt the basic constraints defined first in [11] and generalized in [6] to the generic model describing both TTEthernet and TSN networks following the description from [7].

Frame Constraint. Any frame belonging to a critical stream has to be scheduled between time 0 and its period given the periodic repetition pattern of critical traffic. Hence,

the entire transmission window has to fit within the stream period. To enforce this, we have the condition [7]

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

Link Constraint. Since there can only be one frame at a time on a physical link, we enforce that no two frames that are routed through the same egress port of a device may overlap in the time domain. The constraint [7] is hence

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

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

Stream Transmission Constraint. In order to ensure a low latency, the propagation of frames of a stream must follow a sequential order along the routed path. While this constraint is not necessary for the correctness of the schedule, it ensures that a frame is forwarded only after it has been received by a particular device in order to guide a scheduler towards solutions reducing the end-to-end latency.

Here, one important detail is the network precision, denoted with δ. The precision is the worst-case difference between the local clocks of any two synchronized (e.g. via the IEEE 802.1AS [4] or AS6802 [1] time-synchronization protocol) devices. One difference in the synchonization between TSN and TTEthernet is that the TTEthernet synchronization protocol [15] introduces a delay for critical frames due to the synchronization frames having the highest priority. In the IEEE 802.1AS protocol the synchronization frames have a lower priority than the critical traffic and hence do not introduce a delay. The delay can be accounted for in the scheduling window of the respective frame, i.e., for TSN, the scheduling window only includes the frame duration while in TTEthernet it also includes the worst-case delay of any synchronization frames that might collide with critical traffic.

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

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 ($[v_a, v_x].d$).

End-to-End Constraint. Typically, streams have a maximum allowed end-to-end latency. While this is not always the case, we will define the constraint here for the sake of completeness.

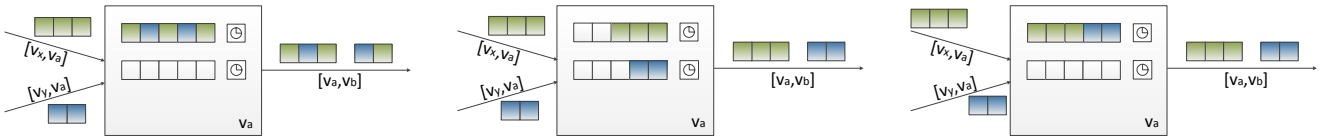
The maximum end-to-end latency constraint specifies that the difference between the reception of a stream at the destination and the transmission of the stream from the sender has to be less than or equal to the specified duration. We denote the sending link of stream s_i with $src(s_i)$ and the last link before the receiving node with $dest(s_i)$. The maximum end-to-end latency constraint [7] is hence

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

B. 802.1Qbv Constraints

The fundamental difference between TTEthernet and TSN networks is that while in TTEthernet the schedule is on the level of individual frames, in TSN the gate control list (GCL) governs the temporal behavior of entire traffic classes (priorities). Nevertheless we want to provide the same degree of determinism on the level of frames in a TSN environment. Hence, we need to derive, in addition to the above mentioned constraints, specific 802.1Qbv constraints. To understand the implications of scheduling on the level of traffic classes we present a simplified example from [7], depicted in Figure 2, where two streams arriving at a switch from different sources are forwarded via the same egress port.

In Figure 2(a), the two streams arrive from different devices at roughly the same time at the represented switch. Due to several factors, like the synchronization imprecision between individual devices or frame loss, the arrival order of frames during runtime can alternate leading to different queue states. Therefore, the order in which the individual frames are placed in the scheduled queue at runtime may be non-deterministic. As noted before, in TSN the schedule controls the opening and closing of the timed gates on the queues of the egress port, not the order of frames in the queue. If the schedule opens the gate for the respective queue first for the duration of transmitting 2 frames and some time later for the duration of transmitting 3 frames, as depicted in the example, there can be any combination of the respective frames of both streams on the egress port at runtime. Hence, the timely behaviour of the two streams may be different during runtime violating the low jitter and end-to-end transmission constraint. One possibility is to account for this worst-case delay using methods like network calculus [16, 17] or trajectory approach [18] and then guide the scheduler towards better solutions if the pessimistic analysis on the end-to-end latency does not fulfil the required maximum. However, we want to provide a fully deterministic behavior and do not consider methods like networks calculus in our solution. In order to avoid this delay and jitter we derive in [7] conditions guaranteeing a deterministic order of frames in the queues. This can be enforced either by not allowing streams arriving during interfering intervals to be placed in the same queue (Figure 2(b)), or by allowing them to be placed in the same queue but ensure that the intended order and transmission time for all frames of the streams are preserved (Figure 2(c)).



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

Figure 2: *Stream interleaving and isolation within an egress port of a 802.1Qbv switch. [7]*

To formulate the constraints we start from a simplified case in which there is only one queue in the device intended for critical traffic and later relax this assumption in order to formulate the complete TSN scheduling constraint.

Let $s_i^{[v_a,v_b]}$ and $s_j^{[v_a,v_b]}$ be, respectively, the stream 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. Stream s_i arrives at the device v_a from some device v_x on link $[v_x, v_a]$. Similarly, stream s_j arrives from another device v_y on incoming link $[v_y, v_a]$. If the two streams arrive from the same device, the link constraint in the previous section ensures that they would not overlap in the time domain. Hence we only look at the case in which the streams arrive from two different devices.

In order to isolate streams that are placed in the same queue we must ensure that they are isolated in the time domain on arrival. Constraint (4) enforces a correct ordering of streams, i.e., if a frame of a given stream has entered a queue, no frame of another stream may arrive at the queue until all frames of the previous stream have been fully dispatched. We remind the reader that the first and last frame of a stream, 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

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

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

The stream isolation constraint is restrictive and may decrease the search space for valid schedules although it may be faster to solve for some use-cases (c.f. [7] for an analysis). It is restrictive since an input where a high-rate stream has a period which is equal to or less than the combined transmission duration of all frames of a low-rate stream will not be schedulable with the stream isolation constraint. This is because there is at least one period instance of the high rate stream in which its frames have to interleave with the frames of the low-rate stream. Given the previous isolation constraint, this would not be schedulable if there is only one queue per port, i.e., the streams must be placed in different queues. This counterexample can be generalized for an arbitrary number of queues.

In order to avoid this, we describe in [7] the frame isolation constraint that allows frames interleaving between streams in a queue while in the same time guaranteeing that the order on the output port is deterministic. We do this by enforcing that there are only frames of one stream in the queue at a time, i.e., frames from another stream may only enter the queue if the already queued frames of the initial stream have been serviced. The frame isolation condition [7] is formulated as

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

The above constraints ((4) and (5)) apply to frames in the same queue. However, as mentioned before, the scheduler may choose to place streams in different queues. Hence, the complete constraint [7] for minimum jitter scheduling of high-criticality streams 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 $s_j^{[v_a,v_b]}.p \leq [v_a, v_b].c$ and where $\Phi(s_i^{[v_a,v_b]}, s_j^{[v_a,v_b]})$ denotes either constraint (5) or, if the stream isolation method is used, constraint (4) between streams $s_i^{[v_a,v_b]}$ and $s_j^{[v_a,v_b]}$.

V. SMT-BASED SCHEDULE SYNTHESIS

We now have a complete set of constraints for both TTEthernet and TSN that describe correct schedules for deterministic real-time behavior of streams. There are several ways how to encode these constraints in order to generate a schedule. One way is to use heuristics, as described in [19]. The advantage of heuristics is that the runtime for such NP-complete problems is usually sub-exponential. The drawback

is that it will not search the entire solution space. Hence, for the average case or for very large networks heuristics may provide the suitable approach for finding schedules. Moreover, heuristics can also be combined with optimization criteria to find improved solutions [20].

However, if we want to have an optimal algorithm we must rely on other methods. The idea of using Satisfiability Modulo Theories (SMT) solvers to find schedules for distributed systems was first proposed by Steiner [11]. 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 ($\mathcal{BV}$) [21, 22]. 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 present a suitable use-case for constraint-satisfaction SMT solving in linear arithmetic [23, 24].

The aim of our scheduling algorithm for TTEthernet and TSN networks is to find values for all individual frame offsets (and queue assignments in case of TSN) in each respective egress port of streams 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. For TTEthernet, the offsets represent the dispatching moments in time for individual frames. For TSN, the frame offsets represent the open and close events for the timed-gate of the assigned queue of the frame/stream.

The drawback of using SMT solvers is that for large networks they may take an unrealistic amount of time to solved the scheduling problem. The scalability of the SMT approach depends on several key factors, analyzed in detail in [6]. Several methods exist that can improve scalability for the average case. Steiner [11] for example introduced an incremental backtracking algorithm that attempts to schedule one stream at a time by adding the stream 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 stream 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 streams are scheduled) or an incremental step is deemed unfeasible by the solver. In the latter case, the constraints of the current stream could not be satisfied with the previous context, so the SMT context is backtracked by removing the last scheduled stream 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 streams 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 streams. Hence, even if the whole set of streams was not schedulable, we can still provide a partial solution for a subset of the given streams.

A. Optimization

While finding a schedule is sufficient in most cases, sometimes the best solution with respect to some optimality criteria is preferred. In most use-cases, the optimality criteria is to minimize the end-to-end latencies of streams. In recent years, a new field, called Optimization Modulo Theories (OMT), has emerged where certain SMT solvers are augmented with optimization capabilities [25, 26, 27]. Alternatively, MIP solvers, like Gurobi [28] or GLPK [29] can be used (cf. [6]). The optimization objective for minimal end-to-end latency is easily expressed based on constraint (3) from Section IV and has been discussed in [6].

For TTEthernet, interesting optimization objectives are related to increasing quality of service metrics for rate-constrained or best-effort streams. We can for example optimize the placement of critical frames in such a way that the bandwidth waste from mechanisms like guard band is minimized. Moreover, we can also optimize for memory overhead in devices by minimizing the time frames are stored in switches before being forwarded.

For TSN, we have seen that traffic is classified into critical and non-critical and isolated in scheduled and priority queues, respectively. Hence, non-critical traffic may benefit from a larger number of priority queues with regard to their quality of service characteristics. However, we still want to find a solution for the critical streams, allowing them to use as many queues as necessary. Hence, any pre-assignment of queues to traffic classes introduces a trade-off between schedulability of high-criticality streams and timeliness properties and flexibility for non-scheduled traffic. This is a typical optimization problem that can be elegantly solved for TSN networks.

In [7] we proposed an optimization objective to find the minimal number of queues required for scheduled traffic such that a valid solution is still feasible. The optimization condition is easy to express using our approach by having an additional variable for each device representing the number of required scheduled queues for high-criticality streams 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 and refer the reader to [7] for the formalization of the optimization objectives.

Here we have to note that there is a design space exploration aspect in the sense that in case of a non-schedulable input, this optimization objective can be used to answer the question on how many queues would be necessary in each device to find a solution (c.f. [7]).

As depicted in Figure 3, the schedule for the remaining priority (non-scheduled) queues can be computed via the inverse of the combined schedule of all scheduled queues. All non-scheduled queues share the same gate open and gate close events and rely on priorities for the transmission order. One drawback is that the resulting scheduling intervals left for BE traffic may be too small to transmit frames of certain

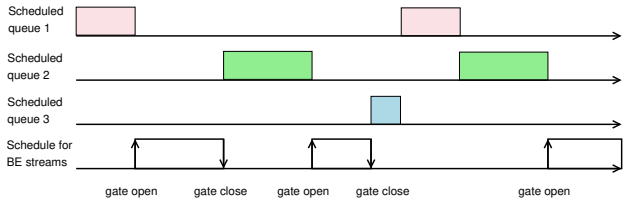


Figure 3: *Example of generating BE-queue schedules.*

sizes (a similar problem to the *checkerboard* fragmentation problem in memory allocators). Hence, an optimization objective worth exploring is maximizing continuous intervals of bandwidth available for non-critical traffic. Taking inspiration from memory management algorithms (c.f. [30]), the optimization objective tries to maintain the *time-space* as non-fragmented as possible, i.e., pack all critical schedule traffic in as few contiguous blocks as possible while satisfying all other correctness constraints. The minimization objective can be formulated as minimizing the accrued sum of inter-event gaps on the combined schedule. In particular, since the constraints defined in Section IV prevent individual frames on the same link from overlapping, this results in minimizing the accrued difference of any two frame offsets on a given link until the hyperperiod.

VI. Conclusion

In this overview based on our previous work [6, 7] we discuss the scheduling problem arising from time-sensitive network technologies like TTEthernet and TSN. We describe the main differences between the two technologies together with generic and specific scheduling constraints that enable real-time temporal behavior on the level of individual communication streams. Based on these constraints, we specify how to encode and solve the problem within the context of SMT solvers, providing also a discussion on possible directions for optimization.

References

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

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

[3] G. Prytz, "A performance analysis of EtherCAT and PROFINET IRT," in *Proc. ETFA*. IEEE Computer Society, 2008.

[4] 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.

[5] 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.

[6] S. S. Craciunas and R. Serna Oliver, "Combined task- and network-level scheduling for distributed time-triggered systems," *Real-Time Systems*, vol. 52, no. 2, pp. 161–200, 2016. [Online]. Available: http://dx.doi.org/10.1007/s11241-015-9244-x

[7] S. S. Craciunas, R. Serna Oliver, M. Chmelik, and W. Steiner, "Scheduling real-time communication in IEEE 802.1Qbv Time Sensitive Networks," in *24th International Conference on Real-Time Networks and Systems (RTNS)*. ACM, 2016.

[8] R. Zurawski, *Industrial Communication Technology Handbook, Second Edition*, ser. Industrial Information Technology. Taylor & Francis, 2014.

[9] W. Steiner and B. Dutertre, "Automated formal verification of the TTEthernet synchronization quality," in *NASA Formal Methods*, ser. Lecture Notes in Computer Science. Springer, 2011, vol. 6617.

[10] H. Kopetz and G. Bauer, "The time-triggered architecture," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 112–126, 2003.

[11] W. Steiner, "An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks," in *Proc. RTSS*. IEEE Computer Society, 2010.

[12] Z. Hanzalek, P. Burget, and P. Šucha, "Profinet IO IRT message scheduling," in *Proc. ECRTS*. IEEE, 2009.

[13] S. S. Craciunas, R. Serna Oliver, and V. Ecker, "Optimal static scheduling of real-time tasks on distributed time-triggered networked systems," in *Proc. ETFA*. IEEE Computer Society, 2014.

[14] J. Forget, E. Grolleau, C. Pagetti, and P. Richard, "Dynamic priority scheduling of periodic tasks with extended precedences," in *Proc. ETFA*. IEEE Computer Society, 2011.

[15] W. Steiner and B. Dutertre, "The TTEthernet synchronisation protocols and their formal verification," *Int. J. Crit. Comput.-Based Syst.*, vol. 4, no. 3, 2013.

[16] F. He, L. Zhao, and E. Li, "Impact analysis of flow shaping in ethernet-avb/tsn and AFDX from network calculus and simulation perspective," *Sensors*, vol. 17, no. 5, p. 1181, 2017.

[17] L. Zhao, P. Pop, Q. Li, J. Chen, and H. Xiong, "Timing analysis of rate-constrained traffic in ttethernet using network calculus," *Real-Time Systems*, vol. 53, no. 2, pp. 254–287, 2017.

[18] H. Bauer, J. Scharbarg, and C. Fraboul, "Improving the worst-case delay analysis of an AFDX network using an optimized trajectory approach," *Industrial Informatics, IEEE Transactions on*, vol. 6, no. 4, 2010.

[19] P. Pop, M. Lander Raagaard, S. S. Craciunas, and W. Steiner, "Design optimization of cyber-physical distributed systems using IEEE time-sensitive networks (TSN)," *IET Cyber-Physical Systems: Theory and Applications*, vol. 1, no. 1, pp. 86–94, 2016.

[20] M. L. Raagaard, "Algorithms for the optimization of safety-critical networks," Master's thesis, DTU, 1 2017.

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

[22] R. Sebastiani, "Lazy satisfiability modulo theories," *JSAT*, vol. 3, no. 3-4, pp. 141–224, 2007.

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

[24] E. Abraham and G. Kremer, "Satisfiability checking: Theory and applications," in *Proc. SEFM*, ser. LNCS, vol. 9763. Springer International Publishing, 2016.

[25] N. Bjørner, A. Phan, and L. Fleckenstein, "νz - an optimizing SMT solver," in *Proc. TACAS*. Springer, 2015.

[26] Y. Li, A. Albarghouthi, Z. Kincaid, A. Gurfinkel, and M. Chechik, "Symbolic optimization with SMT solvers," *SIGPLAN Not.*, vol. 49, no. 1, Jan. 2014.

[27] R. Sebastiani and P. Trentin, "OptiMathSAT: A Tool for Optimization Modulo Theories." in *Proc. CAV*, ser. LNCS, vol. 9206. Springer, 2015.

[28] I. Gurobi Optimization, "Gurobi optimizer reference manual, version 6.0," 2014, retrieved 12-Jan-2015. [Online]. Available: http://www.gurobi.com

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

[30] S. S. Craciunas, C. M. Kirsch, H. Payer, A. Sokolova, H. Stadler, and R. Staudinger, "A compacting real-time memory management system," in *Proc. USENIX Annual Technical Conference (ATC)*. USENIX, 2008, pp. 349–362.