Injection Time Planning: Making CQF Practical in Time-Sensitive Networking

Jinli Yan, Wei Quan*, Xuyan Jiang and Zhigang Sun
College of Computer, National University of Defense Technology
Email: {yanjinli10, w.quan, jiangxuyan, sunzhigang}@nudt.edu.cn

***Abstract*—Time-Aware Shaper (TAS) is a core mechanism to guarantee the deterministic transmission for periodic time-sensitive flows in Time-Sensitive Networking (TSN). The generic TAS requires complex configurations for the Gate Control List (GCL) attached to each queue in a switch. To simplify the design of a TSN switch, a Ping-Pong queue-based model named Cyclic Queuing and Forwarding (CQF) was proposed in IEEE 802.1 Qch by assigning fixed configurations to TAS. However, IEEE 802.1 Qch only defines the queue model and workflow of CQF. A global planning mechanism which maps the time-sensitive flows onto the underlying resources both temporally and spatially is urgently needed to make CQF practical.**

In this paper, we propose an Injection Time Planning (ITP) mechanism to optimize the network throughput of time-sensitive flows based on the observation that the start time when the packets are injected into the network has an important influence on the utilization of CQF queue resources. ITP provides a global temporal and spatial resource abstraction to make the implementation details transparent to algorithm designers. Based on our ITP mechanism, a novel heuristic algorithm named Tabu-ITP with domain-specific optimizing strategies is designed and evaluated under three typical network topologies in industrial control scenarios. Compared with the Naive algorithm without using ITP mechanism, experimental results demonstrate that Tabu-ITP improves the mapped flow number by 10x and the resource utilization by 65%.

***Index Terms*—Time-Sensitive Networking, Resource Mapping, CQF Model, Industrial Control.**

I. Introduction

In many distributed hard real-time and safety-critical application domains, such as automotive and industrial control applications, the current proprietary bus-based networking technologies are reaching their limits in supporting the increasing communication bandwidth requirements [23] [30]. To cope with these emerging requirements, the industry is starting to adopt switched Ethernet as a promising solution. However, the undeterministic queuing delay impedes the switched Ethernet from providing deterministic forwarding service [16] [15]. The deterministic forwarding service has strict requirements on latency, packet loss, and delay variation (jitter) during the packet transmission, which is highly desirable for strict real-time applications [18]. To empower standard Ethernet with such capability, two well-known technologies named Time-Sensitive Networking (TSN) [19] and Time-Triggered Ethernet (TTE) [11] are proposed in recent years.

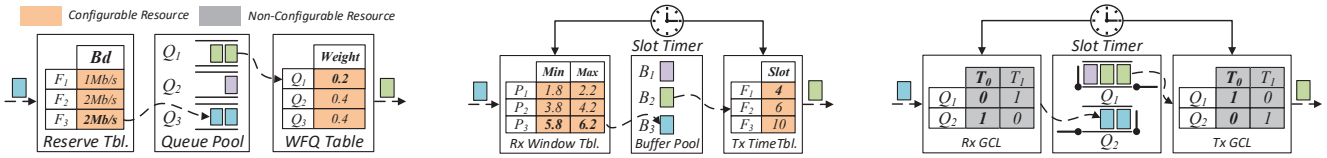
Different from TTE, where the deterministic forwarding service is provided by separated switch buffers with a global conflict-avoiding scheduling mechanism, TSN introduces new traffic shapers like Time-Aware Shaper (TAS) to enable IEEE 802.1 to achieve the same goal [13]. The mechanism adopted in TTE requires to redesign the standard Ethernet switch and generate complex configurations on each switch in the network. This fine-grained scheduling mechanism in TTE faces scalability issues with the growth of demanded flows and network sizes. On the contrary, TSN provides a coarse-grained traffic shaping solution to the deterministic problem with minor extensions to current standard switches. However, the generic TAS still requires dynamic configurations for the Gate Control List (GCL) attached to each queue in a switch.

To simplify the design of a TSN switch, IEEE 802.1 Qch [5] standard recently proposed an easy-to-use model named Cyclic Queuing and Forwarding (CQF) by installing static configurations on the Gate Control Lists (GCL) of TAS. CQF delivers predictable, deterministic latency by cyclically switching the Ping-Pong queues. To the best of our knowledge, by the time of this writing, IEEE 802.1 Qch only defines the queue model and workflow of CQF, and there are only several reviews introducing the related principles [18] [19] [28] [27]. However, how to map the targeting time-sensitive flows onto the underlying hardware resources in CQF-based TSN to make CQF practical is still an open question, which is the primary goal of this paper.

After investigating recently proposed deterministic forwarding techniques, we find that the most obvious difference between TTE and CQF-based TSN is how buffers/queues are used for time-triggered/time-sensitive flows[1]. In TTE, the buffers cannot be shared at the same time slice among time-triggered flows. While in CQF-based TSN, multiple time-sensitive flows can be aggregated into the same queue without considering the input and output time sequences of packets as long as the usage of queues on the path of these flows does not exceed their capacity. Therefore, the complex time scheduling for each time-aware buffer in TTE can be ignored in CQF-based TSN, which significantly simplifies the target problem. We also observe that the injecting time of packets from an end system to the network has a critical effect on the queue utilization of CQF-based TSN switches. If all the packets are transmitted once they are produced, most queue resources will be under-utilized.

*Wei Quan is the corresponding author.

[1]The time-triggered flows and the time-sensitive flows are the same. These flows are scheduled based on per-buffer and per-queue in TTE and TSN respectively.



(a) SRP in standard Ethernet (b) Switch model in TTE (c) CQF in TSN

Fig. 1. Comparison of three typical switch models in standard Ethernet, TTE, and TSN.

Based on these observations, we propose a CQF-oriented planning mechanism named Injection-Time Planning (ITP) for mapping the time-sensitive flows onto the underlying resources both temporally and spatially, which is urgently needed to make CQF practical. ITP provides a global temporal and spatial resource abstraction to make the implementation details transparent to mapping algorithm designers. Therefore, algorithm designers can focus on the problem of maximizing the schedulable time-sensitive flow number by delaying the per-flow sending time on end systems. Based on the ITP mechanism, a novel heuristic algorithm named Tabu-ITP is also proposed to provide a general solver for the aforementioned mapping problem in all possible network topologies. The application-related requirements could be easily satisfied based on our solver without many modifications.

In Tabu-ITP algorithm, we introduce domain-specific knowledge (DSK) to obtain a good trade-off between the mapping quality and the searching overhead. The strategies guided by DSK mainly consist of the flow density and multi-level sorting policy. Flow density is introduced to measure the uniformity of distribution for allocated queue resources, and multi-level sorting policy is designed to determine the flow sequence based on multi-dimensional flow features.

In order to evaluate our proposed solution, we set up three typical topologies in the industrial control network, including ring, linear, and snowflake. Our proposed algorithm Tabu-ITP and other counterparts (Naive, Greedy-ITP, and SA-ITP) are tested under various settings. The experimental results show that Tabu-ITP improves the mapped flow number by 10x and the resource utilization by 65% when compared with the Naive algorithm without ITP. The DSK strategies achieve 39% improvement than random search at most when tested in Greedy-ITP algorithm. Although the mapped flow number of Tabu-ITP and another heuristic algorithm SA-ITP (Simulated Annealing) are almost the same, the Tabu-ITP reduces the time cost by 19% on average.

In conclusion, the main contributions of this paper are:

- We propose an ITP mechanism which maps the time-sensitive flows onto the underlying resources both temporally and spatially to make CQF practical.
- We provide a global resource abstraction to describe the key resource and constraints without exposing irrelevant underlying details to upper mapping algorithm designers.
- A few strategies guided by domain-specific knowledge in CQF-based TSN are concluded to help designers to optimize their mapping algorithms.
- We design a novel heuristic algorithm named Tabu-ITP as a general solver for CQF-based TSN and evaluate the algorithm under various typical topologies and settings.

The rest of this paper is organized as follows. We introduce the motivation in Section II and state the problem in Section III. The algorithm design is proposed in Section IV. Section V and VI present the evaluation results and discussions, followed by the related work and conclusion in Section VII and VIII.

II. Motivation

This section describes the motivation of this work according to our early-stage analysises and investigations of CQF.

A. Features of CQF

In order to illustrate the features of CQF clearly, we compare the representative switch models in standard Ethernet, TTE, and TSN respectively, as shown in Fig. 1. In a Stream Reservation Protocol (SRP) [2] enabled standard Ethernet switch, the queue scheduler plays a key role in flow scheduling for satisfying each flow's SLA (Service Level Agreement) in Fig. 1(a). However, it cannot provide a deterministic forwarding service as the scheduler does not have a precise time control over packets. As can be seen from Fig. 1(b) and Fig. 1(c), the switch models in TTE and TSN are time-centric, which means that the input and output of a packet are strictly time-constrained. In TTE [26], each arriving packet is checked whether it is received within a specified receiving window. And then every legal packet is copied into an exclusive sending buffer for later transmission. When the specified time slot in the TX time table arrives, the corresponding packet would be forwarded without any collisions with other packets. Therefore, the exact sending time slot for each packet on each hop in the target network has to be computed by a global scheduler.

Different from the per-packet scheduling granularity in TTE, the proposed TAS in TSN is at per-queue scheduling granularity. This greatly reduced the scheduling complexity as the number of queues is much smaller than the number of packets in a network. TAS uses time-controlled gates to guarantee a deterministic transmission for each periodic time-sensitive flow. The time-controlled gate executes open/close operations at the start of every time slot on each queue according to the Gate Control Lists (GCL) calculated by a global scheduler. Even though the global complexity of TAS is already simpler than TTE, it still suffers from the complexity and scalability problem.

To simplify the design of a TSN switch, CQF is proposed in IEEE 802.1 Qch standard by installing fixed configurations on the Gate Control Lists (GCL) of TAS. There are two

queues performing en-queue and de-queue operations in a cyclic manner under the control of RX GCL and TX GCL, as illustrated in Fig. 1(c). The reason why CQF could provide the deterministic latency relies on two principles. First, the sending time slot and the receiving time slot of a packet on two adjacent switches must be the same. Second, a packet received at a time slot must be sent at the next time slot in a switch. Thus, the predictable end-to-end latency windows only depend on the time slot size and path length. Compared with the switch model in TTE and TAS, CQF could be easily supported by extending a standard Ethernet switch with statically configured Ping-Pong queues. The main problem for making CQF practical is how to map the time-sensitive flows onto the underlying Ping-Pong queues in switches.

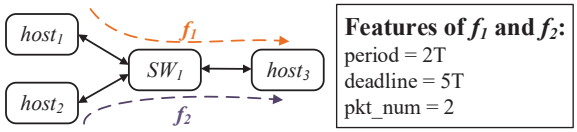
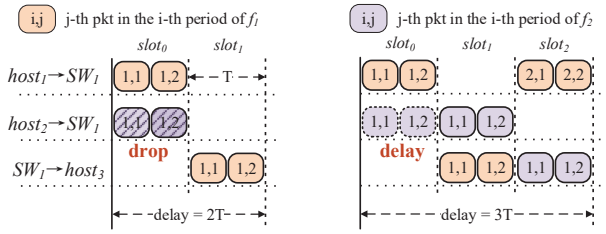


Fig. 2. A simple scenario with two time-sensitive flows.



(a) Before regulation (b) After regulation

Fig. 3. Examples before/after regulating the injection time slot.

B. Observations for CQF Mapping Problem

In most standard Ethernet switches, the packet data is stored in a buffer pool while queues are used to store packets' metadata for scheduling. Thus, the maximal queue length in a switch is equal to the number of packet buffers in the buffer pool. The Ping-Pong queues in a CQF-based TSN switch support both time-division multiplexing[2] and spatial multiplexing. However, there is no need to separately consider the time-division multiplexing problem for each switch under CQF mechanism. According to our early-stage investigations, many periodic time-sensitive flows would easily converge in a part of queues if being forwarded without an elaborated planning of sending time. Since the queue length is limited, many time-sensitive packets will be dropped once a queue overflows. Actually, this problem can be mitigated by delaying the sending time slots of time-sensitive flows with offsets, which makes a more balanced global queue utilization.

To stress it clear, we build a simple scenario where two time-sensitive flows (f_1 and f_2) with a period of "2T" are transmitted by $host_1 - SW_1 - host_3$ and $host_2 - SW_1 - host_3$ separately as depicted in Fig. 2. Each queue can hold two packets at most in a time slot "T". Fig. 3(a) shows that the packets of f_2 will be dropped if the packets of f_1 and f_2 are sent both in slot 0. However, as shown in Fig. 3(b), if we move the sending time of f_2 to slot 1, all the packets can reach $host_3$ successfully. Clearly, there is a higher queue utilization in Fig. 3(b). As the resource utilization improves, the number of schedulable flows grows up.

According to the previous investigations, we can deduce that the mapping between periodic time-sensitive flows and queue resources is determined by the per-flow injection time on end systems in CQF-based TSN. Therefore, in this paper, we propose a mechanism named Injection Time Planning (ITP) to optimize the network throughput for time-sensitive flows by regulating the injection time slots of these flows.

III. Problem Statement

In this section, we build a global resource view and abstract the resource mapping problem based on our ITP mechanism in a CQF-enabled TSN network.

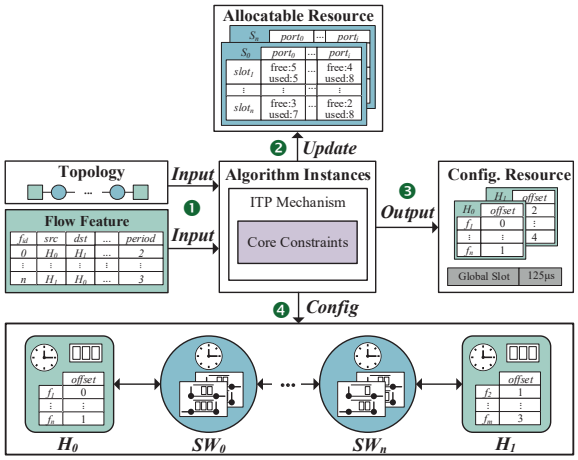


Fig. 4. A global resource view of CQF-enabled TSN network.

A. Resource View in ITP

Based on the CQF-based data plane, a resource view is extracted to decouple the algorithm design from the underlying complex implementation in Fig. 4. The main modules are described as follows.

- **Topology**. It describes all nodes, including switches, end systems, and their connectivities.
- **Flow Feature**. It presents the basic attributes of time-sensitive flows, such as source, destination, period, etc.
- **Allocatable Resource**. It shows the state of each queue resource blocks identified by space (switch and port) and time (time slot).
- **Configurable Resource**. It denotes the configuration information that will be installed into the data plane. Based on our observations, it mainly contains per-flow injection time slots and the global time slot size.
- **Algorithm Instance**. Different algorithms instances can be implemented based on the ITP mechanism. The basic

[2]The global time is divided into multiple equally sized "time slots" for scheduling.

constraints in the following subsection must be satisfied. Besides, users can append application-related constraints.

The entire workflow is described as follows. Firstly, the topology and flow features are loaded into the algorithm (step 1). And then, an ITP-based algorithm instance is performed to plan the injection time slot for each flow under multiple constraints. In this process, the state of the allocatable resource is updated once a flow is mapped (step 2). Finally, the synthesized results are outputted to update the configurable resources (step 3), and these configurations are issued into the hosts and switches in the data plane (step 4).

B. Problem Formalization

We abstract the physical topology as a graph $G = \{V, E\}$. Here V is a set of vertices including the switches and hosts denoted as S and H respectively. E represents a set of directed edges connecting any two vertices.

Since every port in a switch has exclusive CQF queues, the path info should include the outport in each hop. In this paper, the path is described only with the switch nodes that a flow traverses because all allocatable queue resources are in switches. Eq. (1) depicts the entire path where $S_{(1,0)}$ and $S_{(j,k)}$ are the first and last switch info respectively. j and k are the sequence number of switch (starting from 1) and port (starting from 0) respectively.

$$path = (S_{(1,0)}, ..., S_{(j,k)}) \tag{1}$$

The allocatable resources consist of multiple queue resource blocks. The state of each queue block is described as $Q^{T_{(t)}}_{S_{(j,k)}}(free, used)$, where $T_{(t)}$ is the sequence number of time slots, $free$ and $used$ are the number of current free and occupied buffers respectively.

$$\begin{aligned} &\forall f_i \in F, i \in [0, n-1] \\ &f_i = \{src, dst, period, pkt_{num}, deadline, path, offset\} \end{aligned} \tag{2}$$

Here F is the set of time-triggered flows, and the number of flows is n. The feature of each flow consists of source host, destination host, packet number, period[3], and deadline[4]. The path info and the configurable offsets of the injection time slot are at per-flow granularity. Thus, these two factors are appended into flow features, as shown in Eq. (2).

C. Core Constraints in ITP

The regulation of the injection time slot in the proposed ITP mechanism is restricted under multiple constraints to guarantee the mapping validity. Here we illustrate four essential constraints that are irrelevant to specific applications. Users can add extra restrictions according to their requirements.

$$sched_{cycle} = LCM(F.periods) \tag{3}$$

Since time-sensitive flows are transmitted in a periodic pattern, the mapping algorithms need to check whether allocation strategies satisfy the resource constraints within a scheduling cycle. In real scenarios, the periods of flows vary from one application to another. Here the scheduling cycle is equal to the least common multiple of all flow periods in Eq. (3).

1) Offset Constraint: the offset constraint requires that the offset of the injection time slot should be smaller than the flow period. For example, the packets in the first period of a flow have been forwarded before that generated in the second period. There are two reasons for this constraint. First, if this constraint is not satisfied, the packets in multiple periods would occupy the limited storage resource of NIC (network interface card) in hosts simultaneously. Second, this constraint restricts the offset into a reasonable range, which reduces the search space of the per-flow offset.

$$\begin{aligned} &\forall f_i \in F, i \in [0, n-1] \\ &0 \leqslant f_i.offset < \frac{f_i.period}{slot_{cycle}} \end{aligned} \tag{4}$$

where $slot_{cycle}$ is the size of the time slot.

2) Time Slot Constraint: this constraint mainly describes the upper and lower bound of a time slot. In this paper, the granularity of offset is a time slot. It means all periods should be divisible by the pre-defined time slot. Thus, the maximal time slot is the greatest common divisor of the periods set in Eq. (5).

$$max(slot_{cycle}) = GCD(F.periods) \tag{5}$$

With regard to the minimal time slot, it needs to guarantee that all the packets in the CQF queue are transmitted from the upstream node to the downstream node within the same time slot in Eq. (6).

$$min(slot_{cycle}) = \frac{Queue_{size} \times MTU}{B} + hop_{delay} + sync_{prec} \tag{6}$$

where $Queue_{size}$ is the maximum packet amount that each queue can hold, MTU is the maximum packet length. B is the link bandwidth, hop_{delay} is the internal processing delay and propagation delay in a switch and $sync_{prec}$ is the clock synchronization precision.

3) Receiving Window Constraint: the constraint is to check whether a packet arrives at each switch within the right window. In theory, all packets satisfy this constraint if the slot size is not less than the minimal time slot defined in constraint 2. The receiving time slot for each packet in every hop is shown in Eq. (7).

$$\begin{aligned} &\forall f_i \in F, i \in [0, n-1], S_j \in f_i.path \\ &R_{slot}(f_i, S_j) = f_i.offset + hop(S_j, f_i) \end{aligned} \tag{7}$$

where $hop(S_j, f_i)$ starts from 0 and indicates which hop switch j is in the path of flow i.

If this flow is scheduled successfully, the upper bound U_{rw} and lower bound L_{rw} of the receiving window are calculated in Eq. (8).

$$\begin{aligned} U_{rw(f_i,S_j)} &= (R_{slot}(f_i, S_j) + 1) * slot_{cycle} + sync_{prec} \\ L_{rw(f_i,S_j)} &= R_{slot}(f_i, S_j) * slot_{cycle} - sync_{prec} \end{aligned} \tag{8}$$

[3]The time interval taken to produce the specified number of packets.

[4]All the packets in each period must arrive at the destination host before a specific instant of time.

4) Deadline Constraint: in ITP, delaying injection time slot of flow would affect the arrival time at the destination. This constraint is important to restrict all packets of each flow arriving before the specified deadline.

$$\begin{aligned} &\forall f_i \in F, i \in [0, n-1] \\ &f_i.offset + hop_{num}(f_i) \leqslant \frac{f_i.deadline}{slot_{cycle}} \end{aligned} \tag{9}$$

where the $hop_{num}(f_i)$ is the number of switches in the path of flow i.

5) Queue Resource Constraint: the queue resource is split into multiple queue resource blocks according to the combination of $[switch, port, slot]$. The usage of each queue resource block should not exceed the queue length.

First, we set up the mapping between flows and resource blocks. Here the resource block in the kth port of jth switch at the tth slot is denoted as $Q_{S_{(j,k)}}^{T_{(t)}}$. And $S_j.P_{num}$ is the number of ports in the jth switch. If a flow occupies a resource block, the mapping value $M(f_i, Q_{S_{(j,k)}}^{T_{(t)}})$ is 1, as shown in Eq. (10). Otherwise, it is 0.

$$\begin{aligned} &\forall f_i \in F, i \in [0, n-1], \forall j \in [0, m-1] \\ &\forall k \in [0, S_j.P_{num}-1], \forall t \in [0, \frac{sched_{cycle}}{slot_{cycle}} - 1] \\ &M(f_i, Q_{S_{(j,k)}}^{T_{(t)}}) = 1 \\ &\textbf{s.t.} \\ &S(j,k) \in f_i.path, \alpha \in [0, \frac{sched_{cycle}}{f_i.period} - 1] \\ &t = (f_i.offset + \frac{\alpha \times f_i.period}{slot_{cycle}} + hop(S_j, f_i)) \bmod (\frac{sched_{cycle}}{slot_{cycle}}) \end{aligned} \tag{10}$$

where m is the number of switches in the graph. Since the $sched_{cycle}$ is the least common multiple of all the periods of flows, a flow may be transmitted from the host for multiple times. α represents which period the packets belong to. It is used to compute the time slot that the flow occupies within $sched_{cycle}$.

In some cases, the packets forwarded in the first $sched_{cycle}$ cannot arrive at the destination host before the second $slot_{cycle}$ starts. Therefore, the initial resource state in the first $sched_{cycle}$ is different from that in the second $sched_{cycle}$. In order to solve this problem, the time slots that the packets occupy in the first $sched_{cycle}$ are limited within $sched_{cycle}$ by the mod operator in Eq. (10).

$$\begin{aligned} &\forall f_i \in F, i \in [0, n-1], \forall j \in [0, m-1] \\ &\forall k \in [0, S_j.P_{num}-1], \forall t \in [0, \frac{sched_{cycle}}{slot_{cycle}} - 1] \\ &\sum_{i=0}^{n-1} C(i) \times M(f_i, Q_{S_{(j,k)}}^{T_{(t)}}) \times f_i.pkt_{num} \leqslant Queue_{size} \end{aligned} \tag{11}$$

Based on the constructed mapping, the resource constraint is formalized in Eq. (11). We use $C(i)$ to indicate whether the ith flow could be mapped or not. $C(i)$ is 1 if the flow is mapped successfully. Otherwise, it is 0.

In this paper, our optimization objective is to maximize the number of mapped flows, which is expressed as:

$$maximize \sum_{i=0}^{n-1} C(i) \tag{12}$$

IV. Algorithm Design

In this section, we first analyze the problem scale and the choice of search algorithms. Then some optimizations with domain-specific knowledge are introduced to get a good trade-off between the mapping quality and the searching overhead.

The computation of mapping time-triggered flows onto underlying queue resources is equivalent to the bin packing problem and is NP-hard [8]. The search space is $\prod_{i=0}^{n-1} \frac{f_i.period}{slot_{cycle}}$ where n is the number of flows. In order to reduce the complexity while getting a near-optimal solution, we choose computation-intensive heuristics to solve this problem.

The existing heuristic algorithms, such as Simulated Annealing (SA) [10] and Tabu [9], provide generic solutions for the planning problem in many fields. For a specific planning problem, users need to implement customized algorithms by combining these ideas with domain-related features for better solutions. In this paper, we designed a novel Tabu-ITP algorithm based on Tabu search for the advantage of recording the previously visited solutions. And the SA-ITP algorithm is implemented as a counterpart.

However, it is difficult for heuristic algorithms to find a high-quality solution for large scale problems within an acceptable time. We observe that domain-specific knowledge (DSK) could accelerate the entire searching process while guaranteeing the quality of the final solution. Therefore, it is crucial to optimize the generation of candidate solutions by introducing the DSK guided strategies. In this paper, we divide the generation of candidate solutions into two stages. The first stage is to decide the generation mode of candidate solutions. The second stage is to execute the search strategy for candidate solutions under the specified generation mode.

A. Generation Mode

Here we separate the flow set into two subsets: F_{suc} and F_{fail}. F_{suc} is the set of flows that have been mapped successfully while F_{fail} is the set of flows that fail to be mapped. Here we propose two modes to generate candidate solutions by changing the context of current F_{suc} and F_{fail}.

1) Exchanging Mode: in this mode, a set of mapped flows are substituted with a set of unmapped flows, *i.e.*, a part of flows in F_{suc} are moved into F_{fail}. At the same time, the usage of every queue resource block belonging to these flows is cleaned. Subsequently, the flows in F_{fail} are inserted into F_{suc} until the resources are not enough to be allocated.

2) Shifting Mode: different from Exchanging mode, Shifting mode only shifts the injection slots of some flows in F_{suc} for getting a new distribution of resource occupancy. Then, the flows in the F_{fail} attempt to be mapped.

These two modes only define the basic process to generate candidate solutions. In each mode, there are many factors affecting the mapping results, such as which flows in F_{suc}

are substituted/shifted, which flows in F_{fail} are suitable to be inserted into F_{suc} first, and how to determine the injection time slot of each selected flow.

B. Searching Strategy

The searching strategy focuses on how to search for the candidate solution in a specified mode. The search needs to choose a flow and the corresponding offset of the injection time slot. In this paper, we present two searching strategies.

1) Random Search: it selects the flows and injection time slots randomly without any prior knowledge. For example, in Exchanging mode, we randomly remove several flows into F_{fail} at first. Then the flows in F_{fail} are inserted into F_{suc} in random order with random slot selection.

2) Domain-Specific Knowledge (DSK) Guided Search: in particular fields, the domain-specific knowledge is suitable to be introduced for reducing the unnecessary search efforts and getting a near-optimal result. Here we present two DSK-based strategies by combining the flow features and resource state.

Multi-Level Sorting. The flow sequence has an important effect on the mapping results. We observe that the sorting policies based on different flow features result in different results. Thus, we propose a multi-level sorting policy based on the priority order of flow features. The priority is determined by the results under a single flow feature based sorting. In fact, the priority order may be different from scenario to scenario. The sorting strategy for each flow feature is fixed for improving the number of mapped flows. As for packet number and path length, the flow with a larger value should be selected first. With respect to period, the flows with long period should be mapped first because that with low period would appear more often within a $sched_{cycle}$. Finally, the flows with low deadline should be selected first for high probability of success.

$$\forall j \in [0, m-1], \forall k \in [0, S_j.P_{num}-1], \forall t \in [0, \frac{sched_{cycle}}{slot_{cycle}} - 1]$$
$$D = \frac{\sum \left(Q_{S_{(j,k)}}^{T(t)}.used - \frac{\sum Q_{S_{(j,k)}}^{T(t)}.used}{\sum_{j=0}^{m-1} S_j.P_{num} \times \frac{sched_{cycle}}{slot_{cycle}}} \right)^2}{\sum_{j=0}^{m-1} S_j.P_{num} \times \frac{sched_{cycle}}{slot_{cycle}}} \quad (13)$$

Flow Density. This concept is introduced to describe the uniformity of distributions for allocated queue resources. If the injection time slots of flows are regulated randomly, some flows would be aggregated at the same queue resource blocks. It means some resource blocks would be the hotpots so that some flows are impeded from being mapped. Here the variance of usage of global resource represents the flow density (D), as depicted in Eq. (13). If the path is pre-determined, the flow density could be described with the variance of the consumed resource only on the flow path.

3) Probabilistic Selection: in theory, although the DSK guided search could accelerate the convergence of results, the search space is limited. On the contrary, local optima could be avoided with the random search. However, the number of iterations in random search is high because numerous poor solutions would be found. In this paper, the probability coefficient (δ) is introduced to determine the search strategy in every iteration according to a pre-defined probability. This method results in a satisfying trade-off between the overhead (convergence speed) and the quality of mapping results.

Algorithm 1: Tabu-ITP in Exchanging Mode

Input: Flow set: F, Queue resource set: Q
Output: F_{suc}

```
1  init_result, cur_result ← compute_init_result(F, Q);
2  while cur_iteration < max_iteration do
3      cur_num ← 0;
4      while cur_num < max_num do
5          can_result ← cur_result;
6          if random() ≥ δ then
7              dsk_remove_flows(can_result, Q);
8              dsk_insert_flows(can_result, Q);
9          else
10             random_remove_flows(can_result, Q);
11             random_insert_flows(can_result, Q);
12         CAN_set ← can_result;
13         cur_num ++;
14     best_result ← tabu_update_best(CAN_set);
15     cur_result ← tabu_update_cur(CAN_set);
16     if repeat_num ≥ max_repeat then
17         return best_result.F_suc;
18     cur_iteration ++;
19 return best_result.F_suc;
```

With these optimizations, the Tabu-ITP in Exchanging mode is designed in Algorithm. 1. The outer loop counts the iteration number until it exceeds the maximal number. In each iteration, the inner loop generates a set of candidate solutions. In line 6, a random number is generated to determine the search strategy for the new solution. The DSK guided search is executed in line 7-8 while the random search is used in line 10-11. After the candidate solution set is generated, the generic method for updating the best-so-far and current solution in Tabu is performed in line 14 and 15. In order to reduce the superfluous search, the algorithm would be terminated when the repetition of the current best solution exceeds the pre-defined max_{repeat}.

V. Evaluation

According to the described algorithm in Section IV, we implemented Tabu-ITP based on the well-known Tabu search [9]. The Tabu-ITP algorithm is optimized by domain-specific knowledge to achieve a good trade-off between results' quality and overhead.

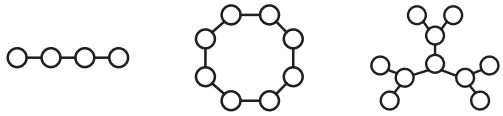
To demonstrate the effectiveness of our proposed algorithm, we compare our Tabu-ITP with three counterparts, including a Naive algorithm, a Greedy-ITP algorithm and a SA-ITP algorithm under different settings in three typical industrial network topologies. The Naive algorithm is to send packets once they are generated without regulating the injection time slot. The other two algorithms are implemented based on ITP mechanism. To be specific, the Greedy-ITP algorithm derives a

mapping based on the greedy policy. The greedy policy refers to the domain-specific knowledge-based search described in Section IV. And the SA-ITP is based on another heuristic search method named Simulated Annealing (SA) [10]. The method to generate candidate solutions in SA-ITP is the same as that in Tabu-ITP.

A. Experiment Setup

All our experiments are conducted on a server with dual Intel Xeon Gold 5120 CPUs (2.2GHz, 56 cores in total) and 128GB of RAM.

1) Topology Selection: the industrial control network is mainly based on the following three topologies: linear, ring, and snowflake, as shown in Fig. 5. Flows in a linear topology-based network can be transmitted in both directions. Differently, flows in a ring topology-based network can only be transmitted in the same direction, *i.e.*, either clockwise or anti-clockwise. A snowflake topology is a specific tree where the number of ports in each switch (except the edge switches) is the same. In our experiments, we set the maximal hop count as 15 in the linear and ring topology and 17 in the snowflake topology.



(a) Linear (b) Ring (c) Snowflake

Fig. 5. Topologies under test.

2) Resource Setting: the resource settings in our experiments include link bandwidth, queue length, and global time slot size. Here, we set the link bandwidth to 1000Mb/s, the length of each CQF queue to 10 and the global time slot size to 125us. The length of each CQF queue is set to 10 for two reasons. First, a time-sensitive packet is not allowed to be cached in a switch for a long time to provide a deterministic forwarding service. Second, the on-chip memory in most TSN switches is limited. As for the global time slot size, we select the minimum time slot, which is 125us according to the Eq. (5), to increase search space of the per-flow injection time slot.

3) Flow features: since there is no open-source flow set available for TSN testing, we refer to the traffic characteristics described in IEC/IEEE 60802 standard for industrial automation networks [1]. All flows in our experiment are generated randomly under the guidance of 60802 standard. Thousands of flows are generated for a better comparison. The packet number in each period for each flow is selected from set $\{1, 2\}$. The period and deadline of each flow are milliseconds in general. We simulate two sets of periods including $\{4, 8\}$ms and $\{2, 4, 6, 8, 10\}$ms. In our experiments, we only generate unicast flows as multicast flows can be split into multiple unicast flows.

B. Experimental Results

Before analyzing the results, we introduce the parameter settings in Tabu-ITP and SA-ITP briefly. Here the size of the tabu list is 500, and the cooling rate in SA-ITP is 0.995. We set the maximum iterations and the maximum repetition to 50000 and 200 respectively for the two terminal conditions in Algorithm 1. In each iteration, 10 candidate solutions are generated. And 5 flows in the set of mapped flows F_{suc} are substituted/shifted for producing a candidate solution. We perform each test 50 times and compute the average of mapped flow number, resource utilization, and time cost.

1) Exchanging mode vs Shifting Mode: the difference between Exchanging mode and Shifting mode is compared under different flow number in Fig. 6. As the number of flows goes up, the improvement from Tabu-ITP and SA-ITP with Exchanging mode becomes larger than that with Shifting mode. The maximal improvement is 28.6% when the flow number is 2000. The reason is that Shifting mode only tunes the injection time of the flows in F_{suc}, where the changes of resource usage are much less than that brought by Exchanging mode. Therefore, in the follow-up experiments, the Exchanging mode is adopted.

2) Random Search vs Domain-Specific Knowledge (DSK) Guided Search: the difference of Random Search and DSK Guided Search is tested in Greedy-ITP Algorithm. The results with different sorting policies and selection policies of injection time in ring topology are shown in Fig 7. The mapping result derived from the flow density-based selection of injection time slot is better than the one derived from the random selection of injection time slot in any sorting policy. The maximum gap between the results derived from the two strategies above reaches up to 17%. In terms of results with different sorting policies, the value from high to low is: path length, packet number, period, and deadline. The multi-sorting policy is based on this order and achieves an improvement of 34.7% than the cases without sorting. While in linear and snowflake topology, the importance of packet number is higher than path length because the average path length in these two topologies is smaller than that in a ring topology. In total, the mapping result with the DSK guided search is 39% higher than that with the random search on average.

3) Effect of Probability Coefficient: the Tabu-ITP and SA-ITP algorithms under different probability coefficient (δ) is tested to obtain a good tradeoff between convergence speed and mapping quality. All candidate solutions are generated based on the DSK guided search when δ is 0. On the other side, all candidate solutions are generated based on the random search when δ is 1. Fig. 8 depicts the mapped flow number and the iteration number under different values of δ. With regard to SA-ITP, the mapped flow number increases from 1060 to 1150 when δ moves from 0 to 0.1. At the same time, the iteration number surges from 234 to 50000 and remains unchanged when δ continue to grow. Since there is only a slight fluctuation in the mapped flow number when δ is greater than 0.1, the δ for SA-ITP is set as 0.1 in the follow-up tests. With regard to Tabu-ITP, as δ increases from 0 to 0.7, the mapped flow number goes up from 1069 to 1142, which is very close to the best results in SA-ITP. Therefore, in the subsequent tests, the δ is set as 0.7 for Tabu-ITP because the

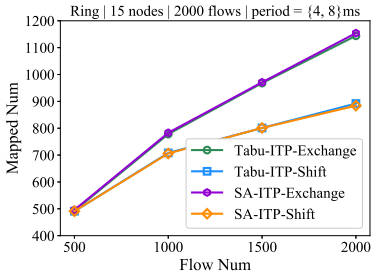


Fig. 6. Exchanging mode *vs* Shifting mode.

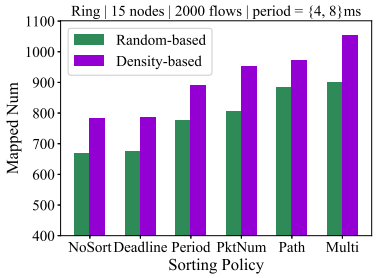


Fig. 7. Random search *vs* DSK guided search.

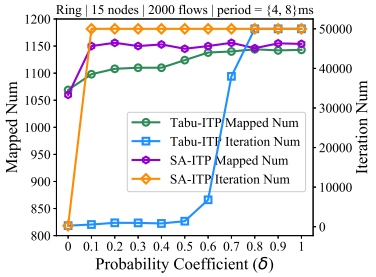
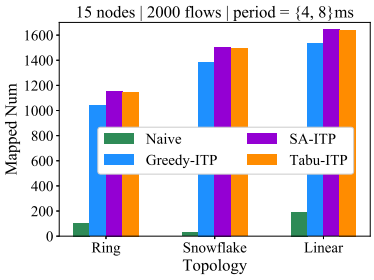
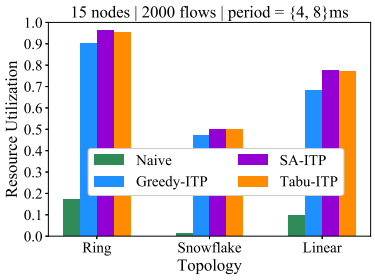


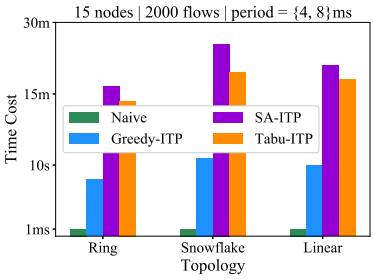
Fig. 8. Comparison under different δ.



(a) Mapped flow number



(b) Resource utilization

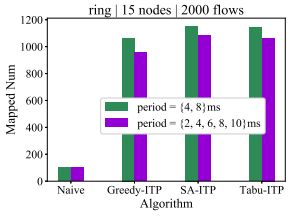


(c) Time cost

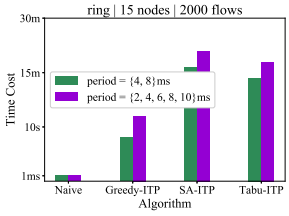
Fig. 9. Comparison of Naive, Greedy-ITP, SA-ITP, and Tabu-ITP algorithms under three different topologies.

extra search is unnecessary when the iteration number increase from 38000 to 50000. The near-optimal result is achieved by introducing a reasonable amount of random solutions. The convergence speed of SA-ITP is slower than Tabu-ITP because the tabu list reduces the repetitions of locally optimal solutions.

4) Comparison of Algorithms with Optimizations: in the above experiments, the most reasonable optimizations are determined. In the next experiments, all the algorithms with these optimizations are compared under different scenarios.



(a) Mapped flow number



(b) Time cost

Fig. 10. Comparison of Naive, Greedy-ITP, SA-ITP, and Tabu-ITP algorithms under different period sets.

First, four algorithms are compared by mapped flow number, resource ratio, and time cost under different network topologies. The successfully mapped flow number using Tabu-ITP and SA-ITP are the highest, which improves the number of the mapped flows by around 10% and 10x compared with Greedy-ITP and Naive algorithms respectively, as plotted in Fig. 9(a). The mapped flow number in the ring topology is the least compared with the other two topologies. The reason is that each switch only has one output port in ring topology and flows are easily aggregated at same resource blocks. The resource utilization is the proportion of the allocated queues over the total queues of all ports, excluding the one connected to the host. Fig. 9(b) depicts that the resource utilization with Naive algorithm is only 10% on average. The Tabu-ITP improves the resource ratio by 65% on average. The resource utilization in Greedy-ITP, SA-ITP and Tabu-ITP are almost the same, and the resource utilization in ring topology reaches up to 96%. It means that the existing best solution is close to the optimal solution. As for the time cost, the Naive and Greedy-ITP algorithm consumes around 1ms and 10s respectively because the computing complexity is very low in Fig. 9(c). The time costs of SA-ITP and Tabu-ITP are over 15 minutes because of the much larger search space. Obviously, compared with SA-ITP, Tabu-ITP reduces the time cost by 19% because the tabu list blocks the existing local optimal results.

Next, four algorithms are compared in ring topology under two different periods set. The first and second period set are $\{4, 8\}$ms and $\{2, 4, 6, 8, 10\}$ms respectively. In Fig. 10(a), the mapped flow number with the first period set is higher than that with the second set because the flows with small period occupy more queue resource than that with large period. In Fig. 10(b), the time consumed under the second set is longer than that with the first period set because the $sched_{cycle}$ for the second set has 960 slots while that for the first set only involves 64 slots.

VI. Discussion

A. Choice of Time Slot

In this paper, the global time slot of a CQF-based TSN switch is set to the minimal time slot for maximizing throughput by improving the frequency for switching the Ping-Pong queues. The smaller the time slot is, the more likely a time-sensitive flow is mapped. However, with the decrease of the time slot, the searching space and therefore the time overhead for each flow grows. This paper focuses on maximizing the

number of time-sensitive flows that can be mapped onto the target CQF-based network at offline stage. While in online flow mapping, it is a totally different scenario. To be more specific, the mapping algorithm must generate new solutions as quick as possible, and the time exhaustive mapping algorithms is no longer accepted. In fact, the number of time-sensitive flows in most scenarios is not very large. Therefore, we would research how to make a good trade-off between the theoretical bandwidth and computing complexity.

B. Path Selection

This paper focuses on the regulation of per-flow injection time. The path for each flow is pre-settled because the path is unique in the topologies under test. In fact, if the path has multiple choices, the successfully mapped flow number will increase with the guidance of flow density. In our next step, the path and injection time slot jointly planning will be researched for achieving better results.

C. Application Constraints

In this paper, we aim to provide a generic mechanism for making CQF practical, and we do not consider the various requirements from upper applications. Here we set all the time-sensitive flows in a standalone mode, and there is no dependence between any flows. In fact, some applications have specific constraints. For example, the packets from flow A should be transmitted before that from flow B. When different users apply our algorithm, it is easy to extend these constraints without much effort.

VII. Related Work

Static Schedule. Steiner *et al.* focus on the static schedule synthesis of time-triggered flows in TTE-enabled network with SMT/OMT solver [24] [21] [22]. However, the computing overhead of SMT/OMT is high in a large-scale network. [24] divides the flow set into multiple subsets. These subsets are then loaded into SMT solver incrementally, and backtracking is used to alleviate conflicts. [22] proposes to isolate these subsets in different search spaces to avoid backtracking. However, some free time slots in a search space could not be used by the flows in other subsets. In this paper, we focus on reducing the computing overhead with domain-specific knowledge, which is orthogonal to the optimizations above.

The schedule in the model of TTE is per-packet while the schedule in TAS [3] is per-queue. [6] observes that when two flows from different links arrive at the same queue, the order in which these frames are placed in the queue is non-deterministic because of synchronization errors, frame loss, etc. To solve this problem, the constraints in TTE are extended with frame isolation restriction. Recently, [20] proposes to formalize the constraints via the first-order theory of arrays. [7] abstracts the schedule in IEEE 802.1 Qbv into no-wait packet scheduling problem, and the algorithm uses Tabu search to reduce the overhead of ILP solver. However, the constraint of queue resource is not analyzed deeply in these papers. We focus on the mapping between the time-sensitive flows and the underlying queue resources from temporal and spatial dimensions.

Dynamic Schedule. With the dynamic change of traffics and topologies, Wan Hai *et al.* observe that it is time-consuming and unpractical to reschedule a large flow set with incremental approaches [29]. They present an algorithm based on mixed-integer linear programming and counterexample resolving process for reacting to the changes fastly. [17] focuses on designing a reconfiguration mechanism to minimize the conflicts of packets during network updates. [14] introduces Software-Defined Networking [12] to synthesize schedules based on the global view. IEEE 802.1 Qcc [4] proposes a centralized model to perform flow mapping with a centralized network configurator (CNC). Our Tabu-ITP and Greedy-ITP algorithm are suitable to be applied in CNC for satisfying static and dynamic mapping requirements respectively.

Mixed-criticality Schedule. In the industrial control network, the flows from different applications are categorized into three types [26]. The priority of them from high to low is: time-sensitive flows, rate-constrained flows and best-effort flows. [26] [25] observe that back-to-back transmission of time-sensitive flows may lead to substantial delays for rate-constrained frames. They proposed to insert blank intervals for reducing the delay of rate-constrained flows without violating the constraints of time-sensitive flows. In the CQF model [5], when all the time-sensitive packets in CQF queue has finished before time slot switching, the left time could be utilized to transmit other flows. In the future, how to support mixed flows efficiently in CQF model would be an interesting direction.

VIII. Conclusion

In this paper, we propose a global planning mechanism ITP for making CQF practical in TSN. The core idea of ITP is to map the time-sensitive flows onto the underlying CQF queue resources efficiently by regulating the per-flow injection time slot. With ITP mechanism, we set up a global resource view to provide a high-level abstraction of resource mapping problem for upper algorithm designers. Then a novel heuristic algorithm named Tabu-ITP is implemented under the guidance of domain-specific knowledge in CQF-based TSN. Finally, three typical topologies in the industrial control network are used to evaluate our work. The experimental results demonstrate that Tabu-ITP improves the mapped flow number and resource utilization by 10x and 65% respectively when compared with the Naive algorithm without ITP. Compared with SA-ITP, Tabu-ITP reduces the computing overhead while mapping almost the same number of time-sensitive flows.

Acknowledgment

This work is supported by National Key Research and Development Program of China (Grant No.2018YFB1800505, 2018YFB1800402), National Natural Science Foundation of China (Grant No.61702538, 61802417, 61601483), Training Program for Excellent Young Innovators of Changsha (Grant No.kq1905006) and Research Project of National University of Defense Technology (Grant No.ZK17-03-53, ZK18-03-40).

REFERENCES

[1] IEC/IEEE 60802 TSN Profile for Industrial Automation. https://1.ieee802.org/tsn/iec-ieee-60802.

[2] IEEE 802.1Qat Standard. www.ieee802.org/1/pages/802.1at.html.

[3] IEEE 802.1Qbv Standard. http://www.ieee802.org/1/pages/802.1bv.html.

[4] IEEE 802.1Qcc Standard. https://1.ieee802.org/tsn/802-1qcc/.

[5] IEEE 802.1Qch Standard. https://1.ieee802.org/tsn/802-1qch/.

[6] Silviu S Craciunas, Ramon Serna Oliver, Martin Chmelík, and Wilfried Steiner. Scheduling Real-Time Communication in IEEE 802.1 Qbv Time Sensitive Networks. In *International Conference on Real-Time Networks and Systems*, pages 183–192, 2016.

[7] Frank Dürr and Naresh Ganesh Nayak. No-Wait Packet Scheduling for IEEE Time-Sensitive Networks (TSN). In *International Conference on Real-Time Networks and Systems (RTNS)*, pages 203–212, 2016.

[8] Emanuel Falkenauer and Alain Delchambre. A Genetic Algorithm for Bin Packing and Line Balancing. In *International Conference on Robotics and Automation*, pages 1186–1192, 1992.

[9] Fred Glover and Manuel Laguna. Tabu Search. In *Handbook of Combinatorial Optimization*, pages 2093–2229. 1998.

[10] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, 1983.

[11] Hermann Kopetz, Astrit Ademaj, Petr Grillinger, and Klaus Steinhammer. The Time-Triggered Ethernet (TTE) Design. In *International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC)*, pages 22–33, 2005.

[12] Diego Kreutz, Fernando Ramos, Paulo Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-Defined Networking: A Comprehensive Survey. *arXiv preprint arXiv:1406.0440*, 2014.

[13] Sune Mølgaard Laursen, Paul Pop, and Wilfried Steiner. Routing Optimization of AVB Streams in TSN Networks. *ACM Sigbed Review*, 13(4):43–48, 2016.

[14] Kilho Lee, Taejune Park, Minsu Kim, Hoon Sung Chwa, Jinkyu Lee, Seungwon Shin, and Insik Shin. MC-SDN: Supporting Mixed-Criticality Scheduling on Switched-Ethernet Using Software-Defined Networking. In *IEEE Real-Time Systems Symposium (RTSS)*, pages 288–299, 2018.

[15] Ziyang Li, Yiming Zhang, Yunxiang Zhao, and Dongsheng Li. Efficient Semantic-Aware Coflow Scheduling for Data-Parallel Jobs. In *International Conference on Cluster Computing (CLUSTER)*, pages 154–155, 2016.

[16] Ziyang Li, Yiming Zhang, Yunxiang Zhao, Yuxing Peng, and Dongsheng Li. Best Effort Task Scheduling for Data Parallel Jobs. In *ACM SIGCOMM Conference*, pages 555–556, 2016.

[17] Zonghui Li, Hai Wan, et al. An Enhanced Reconfiguration for Deterministic Transmission in Time-Triggered Networks. *IEEE/ACM Transactions on Networking (TON)*, 2019.

[23] Johannes Specht and Soheil Samii. Urgency-based Scheduler for Time-Sensitive Switched Ethernet Networks. In *Euromicro Conference on Real-Time Systems (ECRTS)*, pages 75–85, 2016.

[18] Ahmed Nasrallah, Venkatraman Balasubramanian, Akhilesh Thyagaturu, Martin Reisslein, and Hesham ElBakoury. Cyclic Queuing and Forwarding for Large Scale Deterministic Networks: A Survey. *arXiv preprint arXiv:1905.08478*, 2019.

[19] Ahmed Nasrallah, Akhilesh S Thyagaturu, Ziyad Alharbi, Cuixiang Wang, Xing Shao, Martin Reisslein, and Hesham ElBakoury. Ultra-Low Latency (ULL) Networks: The IEEE TSN and IETF DetNet Standards and Related 5G ULL Research. *IEEE Communications Surveys & Tutorials*, 21(1):88–145, 2018.

[20] Ramon Serna Oliver, Silviu S Craciunas, and Wilfried Steiner. IEEE 802.1 Qbv Gate Control List Synthesis Using Array Theory Encoding. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 13–24, 2018.

[21] Francisco Pozo, Guillermo Rodriguez-Navas, Hans Hansson, and Wilfried Steiner. SMT-based Synthesis of TTEthernet Schedules: A Performance Study. In *International Symposium on Industrial Embedded Systems (SIES)*, pages 1–4, 2015.

[22] Francisco Pozo, Wilfried Steiner, Guillermo Rodriguez-Navas, and Hans Hansson. A Decomposition Approach for SMT-based Schedule Synthesis for Time-Triggered Networks. In *International Conference on Emerging Technologies & Factory Automation (ETFA)*, pages 1–8, 2015.

[24] Wilfried Steiner. An Evaluation of SMT-based Schedule Synthesis for Time-Triggered Multi-hop Networks. In *IEEE Real-Time Systems Symposium*, pages 375–384, 2010.

[25] Wilfried Steiner. Synthesis of Static Communication Schedules for Mixed-Criticality Systems. In *International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops*, pages 11–18, 2011.

[26] Domitian Tamas-Selicean, Paul Pop, and Wilfried Steiner. Synthesis of Communication Schedules for TTEthernet-based Mixed-Criticality Systems. In *International Conference on Hardware/Software Codesign and System Synthesis*, pages 473–482, 2012.

[27] Sivakumar Thangamuthu, Nicola Concer, Pieter JL Cuijpers, and Johan J Lukkien. Analysis of Ethernet-Switch Traffic Shapers for In-Vehicle Networking Applications. In *Design, Automation & Test in Europe Conference & Exhibition*, pages 55–60, 2015.

[28] Daniel Thiele, Rolf Ernst, and Jonas Diemer. Formal Worst-Case Timing Analysis of Ethernet TSN's Time-Aware and Peristaltic Shapers. In *IEEE Vehicular Networking Conference (VNC)*, pages 251–258, 2015.

[29] Ningchen Wang, Qinghan Yu, Hai Wan, Xiaoyu Song, and Xibin Zhao. Adaptive Scheduling for Multi-cluster Time-Triggered Train Communication Networks. *IEEE Transactions on Industrial Informatics*, 15(2):1120–1130, 2018.

[30] Xiangrui Yang, Zhigang Sun, Junnan Li, Jinli Yan, Tao Li, Wei Quan, Donglai Xu, and Gianni Antichi. FAST: Enabling Fast Software/Hardware Prototype for Network Experimentation. In *International Symposium on Quality of Service (IWQoS)*, page 32, 2019.