Configuration and Evaluation of Multi-CQF Shapers in IEEE 802.1 Time-Sensitive Networking (TSN)

Alexandris, Konstantinos; Pop, Paul; Wang, Tongtong

Published in:
IEEE Access

Link to article, DOI:
10.1109/ACCESS.2022.3214007

Publication date:
2022

Document Version
Publisher's PDF, also known as Version of record

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 16 September 2022, accepted 4 October 2022, date of publication 12 October 2022, date of current version 18 October 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3214007

RESEARCH ARTICLE

Configuration and Evaluation of Multi-CQF Shapers in IEEE 802.1 Time-Sensitive Networking (TSN)

KONSTANTINOS ALEXANDRIS[1], PAUL POP[2], (Member, IEEE), AND TONGTONG WANG[3]

[1]Huawei Technologies Duesseldorf GmbH, 40549 Munich, Germany
[2]Department of Applied Mathematics and Computer Science, Technical University of Denmark, 2800 Kongens Lyngby, Denmark
[3]Huawei Technologies Company Ltd., Beijing 100015, China

Corresponding author: Paul Pop (paupo@dtu.dk)

ABSTRACT Time-Sensitive Networking (TSN) is a task group of the IEEE 802.1 standardization working group (WG) developing the IEEE 802.1 TSN communication standards. TSN is developing a ''toolbox'' of many standards to provide support for enabling the separation of critical and non-critical traffic, the timeliness and dependability, i.e., reliability, fault-tolerance, and security, of critical traffic. In this paper we focus on the Cyclic Queuing and Forwarding (CQF) traffic shapers, such as the original CQF, the Cycle Specified Queuing and Forwarding (CSQF) and an extension of those, the so-called Multi-CQF shaper. We define formally the problem of configuring CQF-based networks. We have developed a Constraint Programming (CP) formulation for Multi-CQF, as well as a Simulated Annealing (SA)-based metaheuristic solution. These solutions can also obtain results for CQF and CSQF, which can be seen as a special case of Multi-CQF. The CQF configuration problem is NP-hard. We evaluate our solutions on several test cases and scenarios. The CP formulation can find optimal solutions for small problem sizes but does not scale for realistic test cases. However, SA is able to handle large test cases and to find good quality solutions, making it suitable for both design-time and runtime network configuration. We also present comprehensive evaluation results comparing the CQF-based variants (CQF, CSQF, Multi-CQF) on industrial use cases, contrast them to the Time Aware Shaper (TAS, 802.1Qbv), and discuss their advantages and disadvantages.

INDEX TERMS IEEE 802.1 time-sensitive networking (TSN), cyclic queuing and forwarding (CQF), cycle specified queuing and forwarding (CSQF), multi-CQF, network configuration, routing, queue assignment, combinatorial optimization, simulated annealing, constraint programming.

I. INTRODUCTION

We are at the beginning of a new industrial revolution (Industry 4.0), which has several benefits, such as, increased productivity and flexibility, mass customization, improved product quality, reduced waste and emissions, and will lead to increased innovation and new business models. Industry 4.0 architectures consist of distributed interconnected cyber-physical systems (CPS) that monitor and control physical processes that manage, e.g., automated manufacturing, critical infrastructures, smart buildings and smart cities. The applications in these areas are typically safety-critical and real-time, requiring guaranteed non-functional properties, such as, real-time behavior, reliability, availability, safety, and security.

Regarding the communication infrastructure, today, industry uses mostly proprietary protocols [1] that lock customers into the product portfolio of individual product vendors, impairing interoperability. The well-known networking standard IEEE 802.3 Ethernet [2] meets the emerging bandwidth requirements for safety-critical networks, besides remaining scalable and cost-effective. However, Ethernet is unsuitable for real-time and safety-critical applications [3]. Many extensions, such as EtherCAT [4], Profinet [5], ARINC 664p7 [6], and TTEthernet [7], have been developed and used in the industry. Although they satisfy the timing requirements, they

The associate editor coordinating the review of this manuscript and approving it for publication was Huaqing Li.

are mutually incompatible, and hence, they cannot operate on the same physical links in a network without losing real-time guarantees. However, industry is moving towards using standardized solutions such as IEEE 802.1 Time-Sensitive Networking (TSN) [8], upcoming 5G wireless standards [9] and services [10], see [11] for a broader context, as well as interoperability standards such as OPC Unified Architecture (OPC UA) [12]. The IEEE 802.1 TSN Task Group [8] has been working since 2012 to standardize the real-time and safety-critical enhancements for Ethernet. TSN primarily consists of amendments to the IEEE 802.1Q standard. TSN is quickly becoming the de facto standard in several areas, e.g., industrial, automotive, avionics, space, with a wide industry adoption and several vendors developing TSN switches.

In this paper we are interested in cyber-physical systems implementing Industry 4.0 applications, which use TSN for communication, see [13] for a high-level presentation of TSN. TSN is developing several standards,[1] such as, for timing and synchronization, 802.1AS-2020, for supporting bounded low latency, e.g., Scheduled Traffic (802.1Qbv), Asynchronous Traffic Shaping (802.1Qcr), Credit Based Shaper (802.1Qav), Cyclic Queuing and Forwarding (802.1Qch) and Frame Preemption (802.1Qbu), for high reliability, e.g., Frame Replication and Elimination (802.1CB), Per-Stream Filtering and Policing (802.1Qci), Path Control and Reservation (802.1Qca) and for resource management, e.g., Stream Reservation Protocol Enhancements and Performance Improvements (802.1Qat) dealing also with configuration (802.1Qcc, 802.1Qcp) and Link-local Registration Protocol (802.1CS). TSN-based systems are composed of end stations (ESes) interconnected by switches (SWs) and full-duplex physical links, see subsection II-A for the model of a TSN network.

TSN supports the convergence of multiple traffic types, i.e., safety critical, real-time, and regular ''best-effort'' traffic within a single network, and hence, is suitable for mixed-criticality industrial applications. Depending on the application requirements, different traffic types are operating and different combinations of TSN features have to be used. These combinations will determine how messages are scheduled and will require specific approaches for providing timing guarantees. Messages that require low latency and jitter typically use the Time-Triggered (TT) traffic type, implemented via IEEE 802.1Qbv Enhancements to Traffic Scheduling: Time-Aware Shaper (TAS), which relies on schedule tables, called *Gate Control Lists* (GCLs). These define the exact queue transmission times of frames on every egress port along the route of the respective streams. The schedule tables are synchronized to a global notion of time via clock synchronization IEEE 802.11AS *Clock synchronization*. The worst-case end-to-end delays (WCDs) of TT streams are determined by the GCLs. If the WCD of a stream is smaller or equal to its deadline, we say that the stream is schedulable.

[1]The references for all standards can be found based on their names.

However, other traffic types can also be bounded in latency, such as IEEE Audio-Video-Bridging (AVB). AVB [14] introduces two new shaped traffic classes (AVB Class A and B) and uses the Credit-Based Shaper (CBS) defined in IEEE 802.1BA to prevent the starvation of lower priority streams. The WCDs of AVB streams can be bounded by, e.g., Network Calculus-based timing analyses [15]. Besides 802.1Qbv, 802.1Qav, other shapers such as IEEE 802.1Qcr Asynchronous Traffic Shaper (ATS) can be used for applications that require timing guarantees.

For the industrial domain, which is the focus of this paper, TSN has developed the Cyclic Queuing and Forwarding (CQF) standard IEEE 802.1Qch-2017 that introduces a ''peristaltic shaper'', see [13] for an introduction. CQF is useful for applications that do not require very small latencies and jitter, but which are still real-time and require bounded worst-case latencies. CQF divides the time in cycles, and in each cycle it guarantees the transmission of frames from one hop to the next. The end-to-end delay of a stream is then dependent on the number of hops it traverses, and the routing of streams has to be done such that the bandwidth per cycle is not exceeded. Its shortcomings [16], [17] are that (i) it is difficult to decide on a good cycle time (long cycles lead to long latencies and short cycles lead to higher bandwidth requirements), (ii) it uses two buffers (one receiving, one transmitting) and they cannot be filled and emptied at the same time. Hence, researchers have proposed extensions to CQF, such as Cycle Specified Queuing and Forwarding (CSQF) [18], which considers three buffers instead of two. A combination of layer 2 local area CQF and layer 3 wide area CSQF has been proposed in [19]. Furthermore, Finn [16] has recently proposed Multiple Cyclic Queuing and Forwarding (Multi-CQF) as an extension of CQF (IEEE Std 8021Q—2018 Annex T) and has discussed the advantages and disadvantages of several CQF variants.

A. RELATED WORK

There has been a lot of work on the analysis and configuration of TSN networks. For surveys, the reader is directed to [20], [21]. The problem of finding the routes for AVB flows over TSN-based networks has been addressed in [22]. In [23], the authors propose a solution to determine the routes of TT traffic for TSN-based networks using *Integer Linear Programming (ILP)*. The routing of TT traffic for TSN-based networks was also addressed in [24] and solved using a *Tabu Search* metaheuristic.

The configuration issues of TAS (IEEE 802.1Qbv), e.g., related to the routing and synthesis of GCLs, have been extensively studied in the literature [20], [21]. For example, the problem of scheduling TT traffic has been first addressed by [25] using Satisfiability Modulo Theories (SMT). The same problem has been solved using metaheuristics [26], [27].

The joint problem of routing and scheduling has also been investigated for TSN networks. For example, the routing and scheduling for TT traffic has been considered in [23] and [28]

using ILP. [29]. The combination of TT and AVB traffic types has been addressed in [30], which solves the problem of jointly routing and scheduling for both TT an AVB, i.e., it searches for the network configuration where the timing requirements of both TT and AVB messages are satisfied.

However, there have been no systematic investigation on the configuration of CQF shapers. With CQF, SWs are forwarding messages from their input (ingress) ports to their output (egress) ports. Each egress port has a number of CQF priority queues, where messages are placed before transmission. The most general of the CQF shapers, i.e., Multi-CQF, uses two or more queues per priority to forward the messages, and the each priority level can have its own cycle length, see subsection II-B for an explanation on how Multi-CQF functions. In our work, the mixed-criticality real-time applications are modeled as a set of periodic streams. For each stream, we know its source and destination ESes, its period and its deadline. See subsection II-C for the details of the application model.

The configuration problem for Multi-CQF can be formulated as follows, see its formal definition in section III. As an input to the problem we have a TSN topology and a set of applications modeled as periodic streams. For each stream, we are interested to determine (i) its route from source to the destination and (ii) the assignment of the stream to the CQF queues in each SW along its route. We are interested in solutions such that all the streams are schedulable (their deadlines are satisfied), the link capacities are not exceeded, and the mean worst-case end-to-end latencies of streams is minimized.

We are the first to investigate the configuration of Multi-CQF shapers. The only works so far addressing CQF configuration are: [31], which proposes an ILP formulation for the routing and queue assignment configuration problem in CSQF, and [32], which considers that the routes are given and focuses on the assignment of streams to queues in CQF.

The advantages and disadvantages of CQF shaper variants have been discussed in [16] and contrasted to ATS. The authors of [17] present a *qualitative* comparison of CQF, TAS, ATS and CBS in the context of large-scale deterministic networks. However, none of the works so far present *quantitative* evaluations for CQF. The only work [33] that has a systematic quantitative evaluation of shapers, both in isolation and in combination, does not address CQF shapers.

B. CONTRIBUTIONS

We propose several optimization strategies to solve this Multi-CQF intractable configuration optimization problem: Constraint Programming (CP), which can obtain optimal results but has exponential running times, as well as a Simulated Annealing (SA)-based metaheuristic solution, which does not guarantee finding the optimal solution but in practice has been shown to obtain good quality results in a reasonable time. These solutions can also obtain results for CQF and CSQF, which can be seen as a special case of Multi-CQF.

We present comprehensive evaluation results comparing TAS and several CQF-based variants (CQF, CSQF, Multi-CQF) on industrial use cases. We consider that the cycle lengths, queue and bandwidth allocation to priority levels is given. We assume that the total bandwidth allocated over all CQF priorities cannot exceed the link capacity. In addition, we assume that every cycle is an integer multiple of the next-faster cycle [16].

We conclude with a discussion of the quantitative results and a qualitative evaluation the advantages and disadvantages of CQF shaper variants, contrasting them to TAS. The results and discussion are intended to inform and help practitioners and researchers to select the appropriate shapers for their application area and uses cases.

The paper makes the following contributions:

- We formally define the Multi-CQF configuration problem, covering also the CQF and CSQF variants.
- We propose several solutions to the problem of configuring Multi-CQF (and its variants, CQF and CSQF).
- We present, for the first time to our knowledge, a systematic quantitative and qualitative evaluation of CQF shaper variants.
- We show that Multi-CQF can handle well streams with tight timing constraints, but it leads to large latencies for networks with over 1,500 devices with realistic loads. We advocate for a combination of TAS and Multi-CQF shapers, using TAS only for the most demanding streams.
- Finally, we discuss the importance of optimizing the switch configuration of Multi-CQF, i.e., the cycle lengths and the allocation of priority levels and bandwidth.

II. SYSTEM MODEL

The system model consists of an architecture model, a Multi-CQF switch model, and an application model described in subsection II-A, subsection II-B and subsection II-C, respectively. Table 1 summarizes the notations.

A. ARCHITECTURE MODEL

The network contains several ESes that are connected to each other via network switches SWs and physical links. An ES is either the source (talker) or the destination (listener) of an application stream, whereas a switch forwards the frames of streams.

We model the architecture as a directed graph $G = \{V, E\}$, where V is the set of vertices. A vertex $v \in V$ represents a node in the architecture which is either an ES or a SW. $E \subseteq V \times V$ is the set of links. Nodes have input (ingress) and output (egress) ports. We denote the set of egress ports of a node with $v.P$. A port $p \in v.P$ is linked to at most one other node. The set of edges E represents bi-directional full-duplex physical links. Thus, a full-duplex link between the nodes u and v is denoted with both $\epsilon_{u,v} \in E$ and $\epsilon_{v,u} \in E$; a link is attached to one port of the node u and one port of the node v.

TABLE 1. Summary of notations.

Notation	Definition
G	Network topology graph
V	Set of network nodes (ES & SW)
v	Network device (ES or SW)
$v.P$	Set of egress ports for network device v
p	A port
N	Number of priority queues
N_0	Number of Multi-CQF queues
K	Set of queue priority groups
k	Priority group
H	Hypercycle length
c_k	Cycle at priority group k
$\|\|c_k\|\|$	Cycle length for a cycle c_k
C_k	Set of all the cycles c_k in the hypercycle
$\|C_k\|$	Number of cycles c_k in the hypercycle
Q_k	Set of queues at priority group k
$Q_k.b$	Bandwidth allocated to the set of queues at priority group k
$Q_k.B$	Bandwidth of the set of queues at the priority group k within a cycle c_k
E	Set of links
$\epsilon_{u,v}$	A physical link in the network
$\epsilon_{u,v}.s$	Link bandwidth
$\epsilon_{u,v}.d$	Link propagation delay
$\epsilon_{u,v}.D_k$	Link propagation delay in the cycle domain considering the cycle c_k of priority group k
S	Set of streams
s	A stream
$s.k$	Priority group of stream s
$s.v_s$	Talker node for stream s
$s.v_d$	Listener node for stream s
$s.b$	Size of stream s
$s.t$	Period of s
$s.d$	Deadline of s
$s.A$	Arrival pattern of stream s
$s.D$	Cycle-domain deadline of stream s
R_s	Set of all possible routes for s
r_s	Route for stream s
r_s^l	l^{th} link assignment of route r_s
$r_s^l.\varepsilon$	Associated link for link assignment r_s^l
$r_s^l.Q$	Associated queue priority group for link assignment r_s^l
$r_s^l.q$	Associated queue number for link assignment r_s^l

Each link $\epsilon_{u,v}$ is characterized by the tuple $\langle s, d\rangle$ denoting the bandwidth $\epsilon_{u,v}.s$ of the link in Mbit/s and the propagation delay $\epsilon_{u,v}.d$ of the link in μs, which depends on the physical medium and the link length.

B. MULTI-CQF SWITCH MODEL

In the introduction we have motivated the use of Multi-CQF [16]. Here we model the details of a Multi-CQF switch needed to formulate our problem. For further details on how Multi-CQF works, the reader is directed to [16] and the relevant standards. CQF is the original peristaltic shaper and uses only two queues. All CQF variants operate based on cycles of fixed length. In odd-numbered cycles, one queue buffers the frames received by the input port, and the other queue transmits the frames buffered in the queue of the last even cycle. In the even cycles, the roles of the queues are reversed. CQF has to account for an end of cycle "buffer dead time". This is a time when no frames should be transmitted to guarantee that the last frame can be received before the cycle's end, see [16] for details. CSQF introduces a third queue, called a "tolerating queue", besides transmitting and receiving queues. The tolerating queue is used for receiving the packets that arrive early due to delay variations. Multi-CQF goes one step further and introduces queues for each stream priority, and each such group of queues can have their own different cycle length, see the following discussion for details.

An example Multi-CQF switch architecture is depicted in Figure 1, where we have 8 queues, from queue 7 to queue 0. Each port $p \in v.P$ is attached to a link originating from the node v. Thus, the link can also be used alternatively to point out the specific port. Each egress port of a Multi-CQF switch has a set of N priority queues (typically 8) out of which N_0 queues are reserved for CQF traffic and form a set K of priority groups. A priority group $k \in K \triangleq \{1, \ldots, |K|\}$ is a grouping of CQF queues that share the same cycle length and are assigned a bandwidth fraction of the total link bandwidth. Formally, each priority group $k \in K \triangleq \{1, \ldots, |K|\}$ consists of number $|Q_k|$ of CQF queues Q_k (at least two, where $\sum_{k \in K} |Q_k| = N_0$ holds) and is assigned a bandwidth $Q_k.b$ and cycle length $||c_k||$.

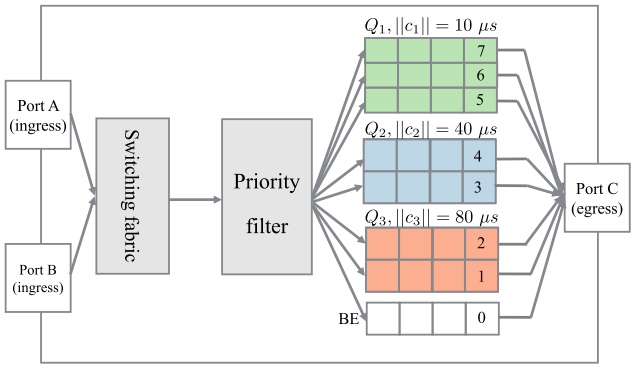

FIGURE 1. Example TSN switch internals with Multi-CQF.

Let us consider the example in Figure 1, where we have 8 queues. Queues 7 to 1 are CQF queues and queue 0 is used for Best Effort (BE) traffic. With our notations, $N_0 = 7$, the number of queues used by CQF out of the total of $N = 8$. In Figure 1 we have three priority groups, i.e., $K = \{1, 2, 3\}$. Priority group 1 has three queues ($Q_1 = \{7, 6, 5\}$) and a cycle length $||c_1|| = 10\ \mu s$, priority group 2 has two queues ($Q_2 = \{4, 3\}$) and a cycle length $||c_2|| = 40\ \mu s$ and priority group 3 has two queues ($Q_3 = \{2, 1\}$) and a cycle length $||c_3|| = 80\ \mu s$.

We define the length of the hypercycle H as the time period after which the network behavior is cyclic, i.e., the network behavior is repeated, typically, the least common multiple (LCM) of the stream periods. For each priority group k, we split the hypercycle into several cycles $c_k \in C_k = \{1, \ldots, |C_k|\}$ with the same length $||c_k||$. We denote with C_k the set of all cycles in a hypercycle for the priority group k and with $|C_k|$ the number of such cycles in a hypercycle.

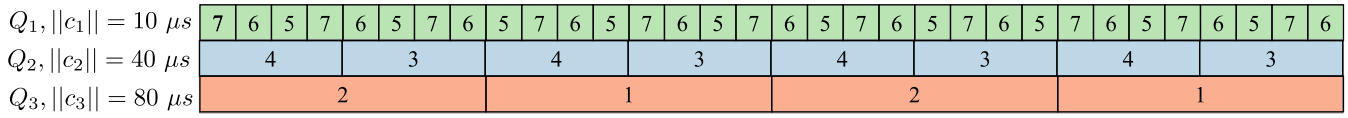

FIGURE 2. Multiple cycles example for Figure 1; the timeline depicts a hypercycle of 320 μs.

The CQF queues in each priority group are controlled by a port control function[2] and are served in a round-robin fashion. Thus, one active queue $q \in Q_k$ is open for transmission and closed for reception, and the $|Q_k| - 1$ inactive queues are only open for reception. The switching fabric receives streams from the ingress ports and forwards each stream to the corresponding egress port and queue, which are decided by our configuration approaches. The number of cycles a stream waits in a queue is *relative* to the cycle it has been received and the cycle it will be forwarded.

However, to simplify the modeling, we use an equivalent *absolute* model, where we assume that the cycle table specifies how many cycles (considering its priority group) each stream should be delayed for transmission relative to the time it has been received in the queue. In our model, we assign a fixed number of cycles to the queues of a priority group, i.e., the queue number 1 is assigned with one cycle, the queue number 2 is assigned with two cycles and the queue number 3 is assigned with three cycles, etc. More generally, the streams assigned to queue q are delayed q cycles. This means that we have to apply a straightforward post-processing step to our *absolute* queue configuration to transform it to a *relative* CQF model that can be loaded in a switch.

Without loss of generality, we assume that (i) the total bandwidth allocated to all priority groups cannot exceed the link bandwidth, (ii) the higher priority groups have a smaller cycle length, and (iii) the starting of cycles at the different nodes is the same and there is no offset. Under these assumptions, every CQF queue will empty before the end of its cycle.[3]

Let us consider an example with 3 streams, with the periods of 80, 160 and 320 μs. In this example the hypercycle length H is the LCM these values, which is 320 μs. Figure 2 shows an example with 3 priority levels, 1 (highest) to 3 (lowest), inspired by [16]. The figure shows a timeline illustrating the cycles. Priority group 1 has a set Q_1 of three queues, 7, 6 and 5 and a cycle length $||c_1|| = 10\ \mu s$. Then, there are $|C_1| = 32$ cycles which have the same length of $||c_1|| = 10\ \mu s$ in the hypercycle vector C_1 for priority group $k = 1$. Priority group 2 has a set Q_2 of two queues, 4 and 3 and their cycle length is $||c_2|| = 40\ \mu s$. Similarly, we have two queues, 2 and 1 at priority group 3, operating with a cycle length $||c_3|| = 80\ \mu s$, double of $||c_2||$. The hypercycle with a length of $H = 320\ \mu s$ is depicted in the figure.

[2]The reader interested in the Per-Stream Filtering and Policy (PSFP) applied to CQF is redirected to Annex T.3 in [34].

[3]See the slides that accompany [16], https://www.ieee802.org/1/files/public/docs2019/df-finn-multiple-CQF-slides-0919-v01.pdf.

C. APPLICATION MODEL

Our application model consists of a set S of real-time streams. Each stream $s_i \in S$ is responsible for sending the frames that encapsulate the data and it is characterized by the tuple $\langle k, v_s, v_d, b, t, d \rangle$ denoting the priority group k of the stream, the source node $v_s \in V$, the destination node $v_d \in V$, the size in bytes, the period in μs and the stream deadline, i.e., the maximum allowed end-to-end delay in μs, respectively.

A stream s is transmitted via a route $r_s \in R_s$, where R_s is the set of all possible routes that the stream can take in the topology graph G from its source to its destination. The route r_s is an ordered list of link assignments, where we denote the l^{th} link assignment with r_s^l, i.e., $r_s \triangleq [r_s^1, \ldots, r_s^{|r_s|}]^T$. The number of link assignments in the route r_s is denoted with $|r_s|$, and it starts from 2 since we assume there is at least one switch in the route. Each route starts with a link assignment originating from the talker $s.v_s$, and ends with a link to the listener $s.v_d$.

Each link assignment r_s^l is characterized by the tuple $\langle \epsilon_{u,v}, k, q \rangle$ denoting the corresponding link, the priority group in the link, and the queue in the priority group, respectively. For example, the second link assignment in the route r_3 is denoted with r_3^2 and characterized by $\langle \epsilon_{5,8}, 1, 2 \rangle$ which are the corresponding link $\epsilon_{5,8} \in E$, the priority group $1 \in K$, and the queue $2 \in \epsilon_{5,8}.Q_1$.

D. CYCLE DOMAIN TRANSFORMATION

Since Multi-CQF is operating using cycles, for modeling purposes, we transform from the time domain into a cycle domain to be able to express, in section III, the governing constraints and objective function. Hence, for a link in a cycle, we define the cycle-domain propagation delay $\epsilon_{u,v}.D_k$ which is equivalent to the propagation delay $\epsilon_{u,v}.d$ of a link $\epsilon_{u,v}$, and the cycle-domain bandwidth $Q_k.B$, which is equivalent to the bandwidth $Q_k.b$ of the queues Q_k at the priority group k. The delay $\epsilon_{u,v}.D_k$ specifies the link delay in cycles of length $||c_k||$ at priority group k and is calculated using $\epsilon_{u,v}.D_k \triangleq \lceil \epsilon_{u,v}.d / ||c_k|| \rceil, \forall k \in K$. The cycle-domain bandwidth $Q_k.B$ specifies the data size in bytes that the queues Q_k at the priority group k can use to transfer data in a cycle c_k and is defined as $Q_k.B \triangleq Q_k.b \times ||c_k||$. Note that all switches and links use the same queue configuration, hence the aforementioned properties of priority groups and their queues are system-wide and not link-dependent.

Similarly, we define in Equation 1 the arrival pattern A, which is equivalent to the size and period of a stream, and cycle-domain deadline $s.D$ which is equivalent to the deadline $s.d$ of a stream s. The arrival pattern A specifies the data in bytes that is sent from the talker in each cycle, considering the cycle length $||c_k||$ of the stream s priority group k:

$$s.A(c_k) = \begin{cases} s.b & \text{for } c_k \times ||c_k|| \mod s.t = 0, \\ 0 & \text{for } c_k \times ||c_k|| \mod s.t \neq 0. \end{cases} \quad (1)$$

The cycle-domain deadline $s.D$ specifies the maximum allowed end-to-end delay in cycles and is defined as $s.D \triangleq s.d/||c_k||$.

III. PROBLEM FORMULATION

Using the models introduced in the previous section, our problem is formulated as follows.

$$\min_{r_s \in R_s} \frac{1}{|S|} \sum_{s \in S} E2E\,(r_s)\,, \quad (2)$$

$$\text{s.t. } C1: E2E\,(r_s) \leq s.D, \quad (3)$$

$$C2: BU\,(r_s) \leq Q_k.B, \quad (4)$$

$$\forall\, r_s \in R_s,\ c_k \in C_k,\ k \in K,\ s \in S.$$

Given the set of all streams S in the system, the network graph G, we are interested in the routing of streams, the assignment of streams to queues, such that all the streams in the system are schedulable, i.e., their deadlines are satisfied (C1 captured by Equation 3), the link capacity constraints are satisfied (C2 given by Equation 4) and the mean end-to-end delay of all streams as defined by Equation 2 is minimized. Let us now present in detail the optimization problem.

A. OBJECTIVE FUNCTION

The objective function is to minimize the mean end-to-end delay of all streams, see Equation 2. The $E2E$ notation represents the worst-case end-to-end delay[4] of each stream and is captured by Equation 5:

$$E2E\,(r_s) \triangleq \sum_{l=1}^{|r_s|} (r_s^l.\varepsilon.D_k + r_s^l.q). \quad (5)$$

The end-to-end delay is the sum of the cycle-domain propagation delay (how many cycles it takes to propagate the stream bits along the link medium) and the cycle shift (how many cycles the stream is delayed in the source node of the link). Assuming that each stream s will be transmitted via the route r_s, each link assignment r_s^l carries link delay $r_s^l.\varepsilon.D_k$ and the associated queue $r_s^l.q \equiv q \in r_s^l.Q \equiv Q_k$, which captures the number of cycles the stream is delayed. Note that a stream s has an associated priority group $s.k$ and hence it can only be assigned the set of queues Q_k of that priority group k. In addition, all queues $q \in Q_k$ have the same cycle length c_k. Thus, using the cycle shift and delay for each link assignment in r_s, we determine the worst-case end-to-end delay $E2E(r_s)$ of the stream s.

[4]We use the terms "delay" and "latency" interchangeably in the paper.

B. CONSTRAINTS

The constraints are defined as follows.

C1: This constraint imposes the restriction that all streams in the application model must meet their deadlines. This constraint iterates over all streams $s \in S$, determines the end-to-end delay of each stream (denoted with $E2E(r_s)$, considering the route r_s of the stream) and checks that this delay does not exceed the stream deadline $s.D$, all defined in the cycle domain.

C2: This constraint enforces solutions to meet the priority groups' cycle-domain bandwidth limits. Each queue at a priority group in the architecture must only transmit less amount of data than its bandwidth in each cycle. For each priority group, the constraint is imposed on the streams that are transmitted via routes that include the link and the priority group. Furthermore, the arrival function of these streams should be shifted to the time when they are queued.

In more detail, this constraint computes the consumed bandwidth of each priority group $k \in K$ in each link $\epsilon_{u,v} \in E$ in each cycle $c_k \in C_k$, which is denoted with $BU\,(r_s)$ and checks for the consumed bandwidth not to exceed the assigned priority group cycle-domain bandwidth $Q_k.B$, all in the cycle domain:

$$BU\,(r_s) \triangleq \sum_{s \in P_m} s.A\,(c_k - T\,(r_s))\,, \quad (6)$$

$$P_m \triangleq \{S \mid r_s^m.\varepsilon \equiv \epsilon_{u,v} \wedge r_s^m.Q \equiv Q_k \wedge r_s^m.q \equiv q \in Q_k \wedge u, v \in V\}, \quad (7)$$

$$T\,(r_s) \triangleq \sum_{l=1}^{m-1} (r_s^l.\varepsilon.D_k + r_s^l.q) + r_s^m.q, \quad (8)$$

$$\forall\, r_s \in R_s,\ c_k \in C_k,\ \forall k \in K,\ m \in M \triangleq \{1, \ldots, |r_s|\}\,.$$

The consumed bandwidth $BU\,(r_s)$ in each cycle c_k considering a priority group k is defined as the sum of sizes of streams in the set P_m that are passing through a particular link in the cycle c_k via the priority group queues. To this end, first we find the streams that are using specific priority group queues in a link for transmission and then determine the latency $T(r_s)$ that takes for each stream s start its transmission on the considered link within its route r_s. We use the arrival pattern function $s.A(c_k)$ and the latency $T(r_s)$ for shifting the pattern to find the stream size at the cycle c_k.

IV. OPTIMIZATION STRATEGIES FOR MULTI-CQF

The problem defined in section III is NP-hard, as proven in [31]. In this section we present the optimization strategies we have developed to determine configuration solutions. We start with a Constraint Programming (CP) formulation that is able to determine an optimal solution, see subsection IV-A. For NP-hard problems, such as our configuration optimization, researchers have also proposed the use of problem-specific heuristics and metaheuristics [35], as an alternative to exact optimization methods such as CP that have exponential running times. Hence, we also

present a Simulated Annealing-based metaheuristic in subsection IV-B.

A. CONSTRAINT PROGRAMMING FORMULATION

In this section we present the CP formulation that is used to optimize the configuration for Multi-CQF. Such an optimization strategy, let us call it CP-MCQF, takes as the inputs the architecture and application models and outputs a set of the best solutions found during search.

CP is a declarative programming paradigm that has been widely used to solve a variety of optimization problems such as scheduling and routing [36]. With CP, a problem is modeled through a set of variables and a set of constraints. Each variable has a finite set of values, called domain, that can be assigned to it. Constraints restrict the variables' domains by bounding them to a range of values and defining relations between the domains of different variables.

CP-MCQF visits solutions that satisfy the constraints defined in section III and evaluates them using the objective function defined in Equation 2 to check if the solution is an *improving* solution, i.e., better than the best solutions found so far. By default, the CP solver systematically performs an exhaustive search by exploring all the possibilities of assigning different values to the variables. Although such a search is intractable for NP-complete problems, it guarantees that the solution is optimal if the search terminates.

We define two sets of decision variables for the CP model, which are associated with the stream routes and the stream queue assignments, respectively. Each decision variable is associated with a domain from which the CP solver decides the variable's value. The mapping function $\mathcal{M}$ captures the mapping of streams to the relative routes from their talker nodes to the listener nodes and the assignment to queues. The domain and the co-domain of the function are defined in Equation 9, where the function domain is the set of all streams in the application model and the function co-domain is the set of all routes from the streams' talker nodes to the listener nodes, captured by the set R_s, which includes all the queues considering the streams' priority group.

$$\begin{aligned} &\mathcal{M} : X \longrightarrow Y, \\ &\text{where } X = \{s | s \in S\},\ Y = \{r_s | r_s \in R_s\}. \end{aligned} \tag{9}$$

The decision variables for stream route and queue assignments, and their domains are defined as $\forall\ r_s \in R_s, r_s^l.Q \equiv Q_k, r_s^l.q \equiv q \in Q_k$.

B. SIMULATED ANNEALING METAHEURISTIC

Several metaheuristic approaches have been presented in the literature [35], and the challenge is to identify the right metaheuristic for our problem. Metaheuristics aim to find a good quality solution in a reasonable time but do not guarantee that an optimal solution will be found. Based on the review of the related work, we have decided to implement a Simulated Annealing (SA) metaheuristic that has been shown in the literature to be a promising approach for routing problems.

Algorithm 1 SA-MCQF($G, S, sol_0, t_0, cr, k_{paths}$)

```
1:  t ← t_0
2:  sol* ← sol ← sol_0
3:  while termination criteria not satisfied do
4:      sol' ← GenerateNeighbor(G, S, sol, k_paths)
5:      if Obj(sol') < Obj(sol) then
6:          sol ← sol'
7:          if Obj(sol') < Obj(sol*) then
8:              sol* ← sol'
9:          end if
10:     else if e^((Obj(sol)−Obj(sol'))/t) > rnd[0, 1] then
11:         sol ← sol'
12:     end if
13:     t ← t · (1 − cr)
14: end while
15: return sol*
```

SA-MCQF is presented in 1, and it takes as input the topology graph G, the streams S, the initial solution sol_0 that acts as the starting point of the search, the initial temperature t_0, the cooling rate cr that controls the temperature decay, and a parameter k_{paths} that specifies how many "shortest paths" according to Yen's algorithm [37] to be considered for routing each stream s. That is, R_s is restricted to these shortest paths for s instead of considering all possible paths between the talker and listener of s. SA-MCQF outputs the best solution sol^* found once a termination condition has been reached (line 3).

For the initial solution sol_0 we use the shortest path for each stream (using Dijkstra's algorithm) and we assign the streams randomly to queues in their priority group. We use a time limit as the termination criterion. SA is a variant of the neighborhood search technique [35], where the local search space is explored via the generation of a new solution sol' starting from the current one sol (line 4). The neighbor solutions sol' are generated through performing design transformations (also called *moves*) on sol. In general, the new solution is accepted if it is improving the objective function (line 6). SA records the best-so-far solution found (line 8) to return it at the end (line 15). However, the key aspect of SA, which helps it to avoid getting stuck into local optima, is that a worse solution can also be accepted with a certain probability (line 10, where *rnd* returns a random number in [0, 1], with a uniform distribution). The basic concept of SA is inspired by the slow cooling of solid material in a heat bath, known as *annealing*. The cooling rate affects the properties of the cooled material. Hence, the probability of accepting a worse solution depends on the deterioration of the objective function and on a *cooling scheme* captured by a control parameter called initial temperature, t_0, which is analog to the temperature concept of the physical annealing process and a cooling rate cr, which is the rate at which the temperature drops with time.

We use the objective function from Equation 2 in our SA, with the difference that we normalize the worst-case end-to-end delays and we penalize unschedulable and invalid solutions. We allow SA to visit unschedulable and invalid solutions, i.e., solutions where constraints C1 in Equation 3 (meeting all deadlines) and C2 in Equation 4 (meeting the bandwidth limits) are not satisfied. This is to allow SA to explore areas of the search space that may lead to improved solutions. Thus, in case a solution sol' is not schedulable, i.e., the constraint in Equation 3 is not satisfied, we add a "penalty" value of 2. In case a solution is invalid, i.e., a link capacity constraint from Equation 4 is not satisfied, we penalize the objective function value by adding a penalty value of 3. Since, we are adding the penalties to the objective function for invalid solutions, we normalize $Obj(\cdot)$ to bring it in a similar range as the penalties [38]. Thus, our $Obj(\cdot)$ is the sum of all stream latencies, divided by the total number of streams, where each latency value is normalized by dividing it with the stream's deadline. Note that *Obj* is smaller or equal to 1 if all streams meet their deadlines.

The *GenerateNeighbor* function performs *moves* that change the routing and queue assignment of a stream, returning a neighboring solution sol'. *GenerateNeighbor* selects randomly with a uniform distribution between two moves (1) change route and (2) change queue. For the *change route* move, *GenerateNeighbor* picks a random stream and then picks a random path out of the k-shortest-paths generated by Yen's algorithm [37]. For the *change queue* move, *GenerateNeighbor*, we have the route and we want to modify a queue assignment for one of the links in the route. Hence, we pick a random stream to apply this move and then we pick a random link along its current route. For the egress port corresponding to that link, we change randomly the queue assignment of the stream, keeping the stream in the same priority group.

V. EVALUATION

We have performed several experiments, see the following subsections. The CP formulations have been implemented in C# using Google OR-Tools [39] as the CP solver. The SA for TAS was implemented in C++, whereas the SA for Multi-CQF (SA-MCQF) was done in Python. The termination criteria for both SA implementations were time limits, see respective sections for each experiment. Both the SA parameters and time limits have been determined such that no improvement is seen for longer periods of time. The longer time limit for SA-MCQF has been set to compensate for the slower Python implementation.

We have considered industrial ring topologies that are typical for industrial networks (cf. IEC/IEEE 60802), with 8 to 512 switches, see Table 2. We consider that links have a rate of 1 Gbps and that the propagation delay in the switch is zero. To each switch we connect two end systems, resulting in 16 to 1024 end systems for each test case, respectively. The test cases have an increasing number of streams, from 29 to 1,844, respectively.

We have considered timing constraints which are "relaxed" (corresponding to priority levels 6 and 5 in IEC/IEEE 60802), "tight" (corresponding to priority 7) and "mixed" (a mixture of the relaxed and tight constraints). Thus, for the relaxed test cases we have used periods of 1,000, 2,000, 5,000, and 10,000 μs and randomly generated message sizes between 50–500 bytes (B). For the tight test cases we have used periods of 100, 500, 1,000, 1,500 and 2,000 μs and sizes between 30–100 B. The mixed test cases have half of the streams as tight and the other half as relaxed. For all streams, the deadlines are equal to the periods.

We have used ParamILS [40] to determine the SA parameters, i.e., the initial temperature t_0 and the cooling rate cr. For all SA implementations, the values for t_0 for the different test cases are presented in Table 2, where we also show the time limits in seconds. We have used a value of $k_{paths} = 10$ for all SA runs. All algorithms were run on a MacBook Pro with an ARM-based Apple M1 Pro processor, with 6 performance cores and 2 efficiency cores, and with 32 GB of RAM.

A. COMPARISON OF SA AND CP

In the first experiment we were interested to determine the scalability of our CP and SA implementations for CSQF, and the ability of SA to find good quality solutions. To generate the configurations for TAS we have used the CP-based approach from [41] (let us call it CP-TAS) and the SA-based approach from [30] (let us call it SA-TAS). The CSQF configurations have been obtained with the Multi-CQF approaches we have developed in this work, namely the CP-based approach presented in subsection IV-A and the SA-based approach presented in subsection IV-B, considering the particularities of the three-queue CSQF shaper. These CSQF variants are denoted by CP-CSQF and SA-CSQF.

We have considered test cases number 1 to 3 from Table 2. A cycle length $||c|| = 20\ \mu s$ is used. The CP has been run until completion, which guarantees that the obtained solution is optimal. The first observation is that the CP approach is not scalable, i.e., it can only handle test cases up to 24 switches, and does not terminate (even after running for two days)

TABLE 2. Test cases for the first and second experiments and SA parameters.

Test Case	Switches	End stations	Streams	t_0	cr	Timeout (s)	
						SA-TAS	SA-CSQF
1	8	16	29	0.90	0.05	120	120
2	16	32	58	0.90	0.05	180	180
3	24	48	87	0.90	0.05	180	220
4	32	64	116	0.90	0.04	180	240
5	48	96	173	0.90	0.04	200	440
6	64	128	231	1.00	0.03	240	620
7	128	256	461	1.00	0.03	240	1800
8	256	512	922	1.10	0.02	360	3600
9	512	1024	1844	1.20	0.01	420	4000

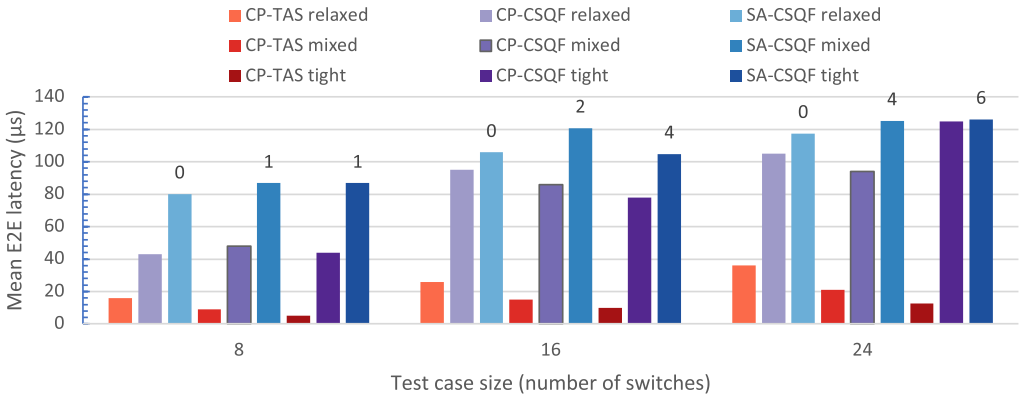

FIGURE 3. Evaluation of SA for TAS and CSQF compared to CP optimal results.

for larger test cases. The CP-CSQF runtime was on average 1,564 s for the three test cases. Therefore, we use CP here only as a way to obtain optimal solutions in terms of mean end-to-end latency, to be used as a baseline for the comparison with the SA approaches, and we do not advocate for its use on realistic test cases.

Figure 3 reports the results obtained. On the y-axis we have the mean worst-case end-to-end latency (henceforth called *Mean E2E*) in μs. On the x-axis we have the three sizes of test cases, with 8, 16 and 24 switches. For each size, we consider relaxed, mixed and tight test cases. The CP-TAS results are plotted with bars using shades of red, and are used as a baseline. We do not plot the SA-TAS results because they are identical to CP-TAS (both algorithms have obtained the optimal results). The CP-CSQF results are depicted with shades of purple and the SA-CSQF with shades of blue.

As expected, TAS always obtains the smallest latencies. TAS has been evaluated in the literature and compared to other shapers [33], and the conclusion is that it provides the lowest latency, jitter and bandwidth usage. This can be seen in the figure, where the red bars (CP-TAS) are smaller than the purple and blue bars. Compared to the optimal CSQF results obtained with CP-CSQF, TAS is able to reduce the mean E2E latencies by 2.69 to 9.99 times. The lowest difference is for relaxed test cases, the largest for tight and the difference for mixed test cases is in-between. In the figure, the darker the shade of color the tighter is a test case in terms of timing constraints. Note that TAS meets all deadlines, whereas CSQF misses a few deadlines due to the larger latencies, see the numbers on top of the bars, which denote the deadline misses. All these deadline misses with CSQF can be avoided by using a cycle length of 10 μs instead of 20, see the next section for a discussion, with the exception of the tight test case with 24 switches, which will still have two missed deadlines out of 87 streams, see Figure 4.

As a final observation for this experiment, we can see that SA-CSQF (blue) is able to obtain results that are not too far from the optimal result, as obtained with CP-CSQF (purple), i.e., they are only 1.01 to 1.97 times larger. This shows that our SA-CSQF can obtain good quality results within a reasonable time, see the time limits in Table 2. This means that SA-CSQF if preferred for configuring CSQF shapers, compared to CP-CSQF, which is not scalable. The near-optimal results obtained by our SA implementation have been determined using the time limits from Table 2. Note that good quality results can be obtained in much shorter times with SA compared to the time limits Table 2, which allows the handling very large realistic test cases.

B. EVALUATION OF CSQF COMPARED TO TAS

In the second experiment, we were interested to evaluate the CSQF shaper behavior as the size of the test cases is increasing. Hence, we have used test cases of increasing size as presented in Table 2. We have used two cycle lengths $||c||$ of 10 and 20 μs. The results are depicted in Figure 4, where we have a similar setup as in the previously discussed Figure 3. All the CSQF results have been obtained with SA-CSQF and the all the TAS results with SA-TAS. With red bars we depict the TAS mean E2E latencies, with purple those for CSQF considering a $||c||$ of 20 and with blue a $||c||$ of 10 μs. The darker the color, the tighter the timing constraints; each size of test cases considers three variants: relaxed, mixed and tight test cases.

As we can see from the figure, SA-CSQF is able to handle large test cases, with topologies of up to 1,536 devices and 1,844 streams. As in the previous experiment, CSQF results in larger latencies. As mentioned, this may lead to deadline misses. Hence, on top of each CSQF bar we also denote the number of missed deadlines, out of the total number of streams, see Table 2, in the respective test case size.

TAS can handle tight streams with no deadline misses. CSQF has larger latencies than TAS because it has to wait 1, 2 or 3 cycle lengths (depending on the queue assignment) to push data to the next hop. Therefore, as expected, if the cycle length decreases from 20 μs to 10 μs, the CSQF latencies are also decreasing, from 9.34 times on average longer than TAS to only 4.85 larger than TAS on average. This means that CSQF can handle small to medium test cases, and can even handle large test cases if the cycle length is small. In the figure, the number of missed deadlines is a more important

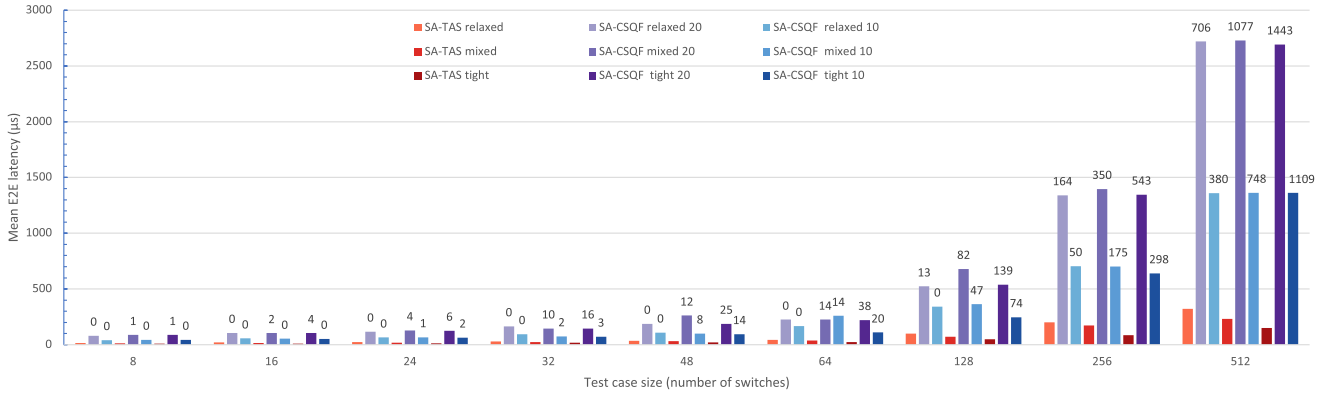

FIGURE 4. Evaluation of CSQF compared to TAS for increasing test case size.

metric than the mean E2E latency, which depends on the characteristics of the streams and the network size, i.e., longer routes will lead to larger latencies. CSQF handles better more relaxed test cases, but the size of the test case is a more important factor. Our SA-CSQF approach optimizes the routing of streams and their assignment to queues such that streams with tighter deadlines will use shorter routes and "faster" queues, whereas streams with more relaxed deadlines may end up being placed in "slower" queues and use longer routes.

As the test case size increases, the network load also increases, from a 1% to 104% maximum link utilization. If a link is over 100% capacity it means that the capacity constraint is not satisfied. That was the case in the relaxed test cases for 256 and 512 switches for $||c|| = 20\ \mu s$ and for 128, 256 and 512 switches for $||c|| = 10$. For the mixed test cases, the test cases with 64, 128, 256 and 512 switches were infeasible for $||c|| = 10$ and only those with 256 and 512 switches remained infeasible when $||c||$ was increased to 20, since increasing the cycle length makes it easier to fulfill the link capacity constraint. For the tight test cases, only the largest test case with 512 switches was infeasible, for both cycle lengths.

For smaller network loads, CSQF can work with small cycle lengths, which reduce the latencies, since the bandwidth capacity constraint in a cycle (see Equation 4) is easier to satisfy. Note that the results in Figure 4 depend on the test cases used for the experiments. In our setup, as mentioned, we have used demanding timing constraints, i.e., periods and deadlines from 100 to 2,000 μs (tight test cases and half of the mixed) and from 1,000 μs to 10,000 μs (relaxed test cases and the other half of the mixed). CQF and its variants are intended to handle streams with more relaxed timing constraints compared to TAS, which is intended for streams that require very low latencies and jitter. Our investigation was aiming to show the limits of CQF, which, as expected, has difficulties with long routes and overloaded networks, since the latencies depend on the number of hops and the cycle length. We explore in the next experiment the impact of the network load on the CSQF ability to handle industrial streams.

C. EVALUATION OF CSQF FOR INCREASING NETWORK LOAD

Our next experiment investigates the ability of CSQF to handle situations, where the network load is increasing. Hence, we have used a topology with 8 switches, and we have increased the number of streams from 29 to 864 streams, see Table 3 for the details, including the SA parameters used. We have considered "mixed" timing constraints for these test cases.

We depict the results in Figure 5 similarly to the previous graphs. We denote SA-TAS mean E2E latency with red and the results of SA-CSQF with purple. We have also obtained results for CQF, which are denoted with a gray bar. The CQF results have been obtained by configuring SA-MCQF to consider only two queues and a "dead buffer time" of 25% of the cycle length, see [16] for a discussion on how "2-buffer CQF" works. For each test case, we have tried cycle lengths from 10 to 120 μs.

As the load increases, it is not possible to obtain feasible solutions that satisfy the capacity constraint in Equation 4, which means that there will be cycles when at least one link is overloaded, that is, the link should carry more than 100% its capacity, which is not possible. Therefore, for each test case, we have identified that cycle length which satisfies the capacity constraint, see the three numbers on the x-axis on top of the number of flows in Figure 5. Determining the cycle length that minimizes the worst-case E2E latencies and at the same time satisfies the capacity constraint is an optimization problem in itself. For our optimization problem

TABLE 3. Test cases for the third experiment, mean link utilization, and time limits. The time limits for SA-CQF are the same as for SA-CSQF.

Test Case	Switches	End stations	Streams	t_0	cr	Cycle length	Mean LU (%)	Max LU (%)	Timeout (s) SA-TAS	Timeout (s) SA-CSQF
10	8	16	144	0.20	0.010	13	3%	87%	120	600
11	8	16	288	0.20	0.008	26	7%	79%	180	1000
12	8	16	432	0.20	0.007	32	9%	88%	180	1000
13	8	16	576	0.30	0.006	44	12%	88%	200	1200
14	8	16	720	0.30	0.005	51	15%	90%	240	1400
15	8	16	864	0.30	0.004	67	18%	88%	320	1600

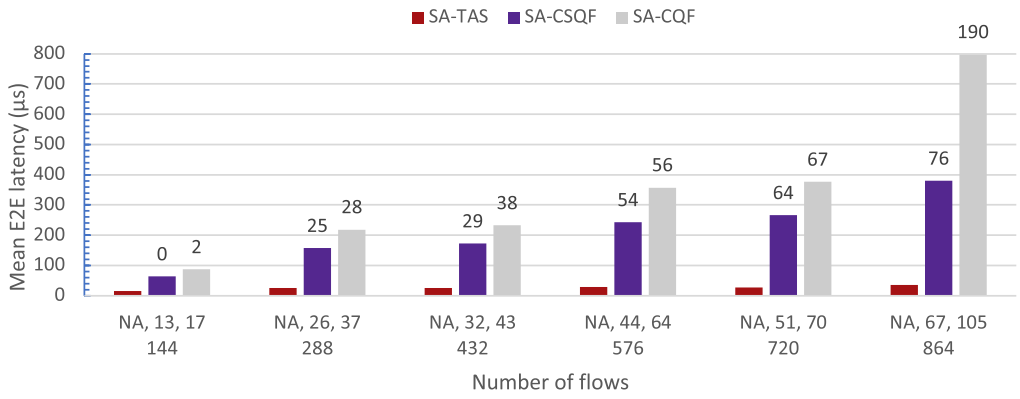

FIGURE 5. Evaluation of CSQF and CQF compared to TAS for increasing network load. The three numbers on the x-axis on top of the number of flows are the cycle lengths ||c|| that have to be used to obtain a feasible solution for the test case, for TAS (NA, because it is not cycle-based), CSQF and CQF, respectively.

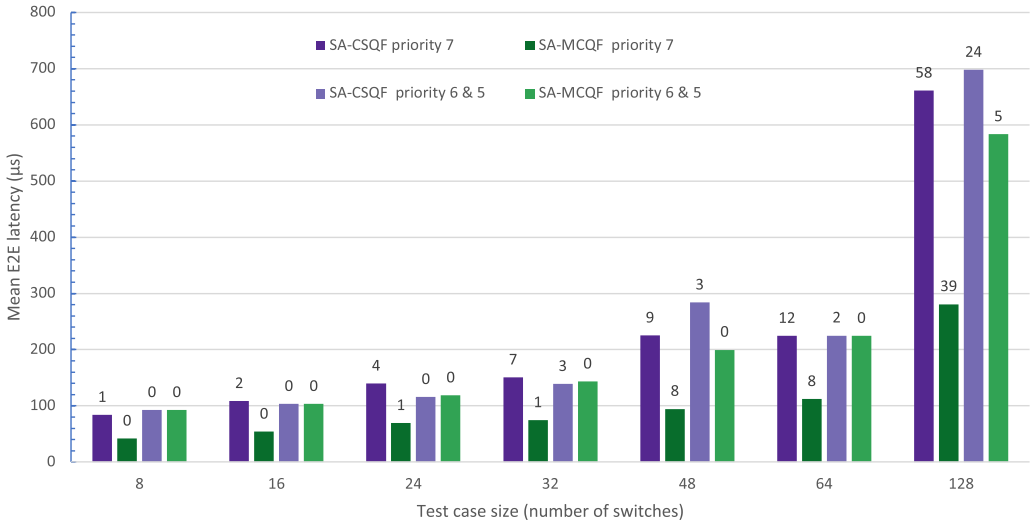

FIGURE 6. Evaluation of Multi-CQF compared to CSQF for ring topologies of increasing size.

we consider that the "switch configuration" is given, i.e., the priority groups, the number of queues in a group, the bandwidth allocations, and the cycle lengths. We leave the optimization of such switch configurations to future work. For the evaluations in this section, we have determined the cycle lengths manually such that the capacity constraints are satisfied (otherwise the solution is invalid) and the mean E2E delay is minimized. Note that for these cycle lengths, although the link capacity constraint is satisfied, not all the deadlines are satisfied in the respective test case. We can see that because of the need to increase the cycle length with CSQF, the mean E2E latencies are also increasing. However, CSQF, which uses three queues, outperforms CQF, which uses only two queues. Thus, CQF results in mean E2E delays that are between 1.35 and 2.10 times larger than those of CSQF. Also, CQF is leading to more deadline misses, see the numbers on top of the gray bars, e.g., deadlines missed for 190 flows out 864 total for the most loaded test case.

The conclusion is that as the load of a network increases, CSQF has to accommodate the increase in bandwidth requirements by increasing the cycle length, leading, therefore, to an increase in latencies and deadline misses, see the numbers on top of the purple bars in Figure 5. CQF experiences larger mean E2E delays compared to CSQF, resulting also in more deadline misses.

D. COMPARISON OF CSQF WITH MULTI-CQF

In this experiment we were interested to evaluate Multi-CQF and contrast it with CSQF. Thus, we have used the "mixed" test cases from Table 2 that were feasible (the bandwidth constraint is satisfied) with a cycle length of 20 μs using CSQF, i.e., from 8 to 128 switches. Larger "mixed" test cases did not have feasible results for $||c|| = 20\ \mu s$, i.e., the link capacity constraint was not satisfied. As mentioned, these test cases use a mixture of priority 7, 6 and 5 streams from IEC/IEEE 60802. Hence, we have designed a switch configuration to assign these two stream priorities to two

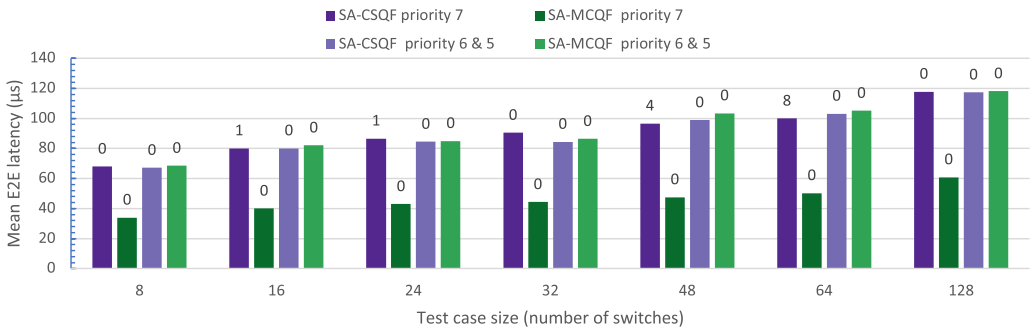

FIGURE 7. Evaluation of Multi-CQF compared to CSQF for Erdős-Rényi topologies of increasing size.

different priority groups. Thus, priority group 1, corresponding to stream priority 7, uses three queues, has a cycle length of 10 μs and an allocated bandwidth of 50%. Priority group 2, used for stream priorities 6 and 5, also has three queues, and 50% allocated bandwidth, but uses a cycle length of 20 μs.

Figure 6 shows the results, where we use purple for CSQF and green for Multi-CQF. The setup is similar to the previous graphs that showed the mean end-to-end latencies. In the figure, we show separately the latencies and missed deadlines for the two stream priorities (priority 7, a higher priority with smaller periods and deadlines, and priorities 6 and 5). As we can see from the figure, Multi-CQF is able to reduce the latencies compared to CSQF. This is because it can tailor the cycle length and bandwidth to the particularities of the streams, i.e., it can use a smaller cycle length for the priority 7 streams with smaller periods and deadlines. Also, the Multi-CQF switch configuration we considered is using more queues compared to CSQF.

In Figure 7 we have done the same evaluation as in Figure 6, but instead of the ring topologies used in Table 2, we have used Erdős-Rényi graph topologies. We have used the same SA parameters as in the previous experiments. The same conclusion holds for these topologies, i.e., Multi-CQF is able to produce solutions where the deadlines are satisfied, compared to CSQF, which misses deadlines. Multi-CQF is able to meet the deadlines of the tight streams (priority 7) at the expense of slightly larger latencies for the relaxed streams (priorities 6 and 5), which however, are still meeting their deadlines. By optimizing the switch configuration, e.g., giving more bandwidth to the relaxed streams, it is possible to trade-off the latencies of tight vs. relaxed streams to tailor their requirements. Note that the Erdős-Rényi graphs are easier to solve, since they have shorter routes on average between talkers and listeners compared to the ring topologies used in Figure 6.

This shows that when carefully deciding on the cycle lengths, number of queues and bandwidth allocation for the priority groups, Multi-CQF is capable of handling industrial streams of mixed requirements. However, as the network diameter increases, all the peristaltic shapers (CQF, CSQF, Multi-CQF) experience increase latencies because the latency of a stream depends on the number of hops and cycle length. For realistic network loads, the cycle lengths cannot be reduced too much because this leads to situations, where the link capacities are not satisfied (they are used over 100% capacity). Only Multi-CQF has the option to allocate more bandwidth for the demanding streams that require smaller latencies, at the expense of the other, less demanding, streams in the network. And, even for small cycle lengths, if the routes are long, the latencies will be long. This shows that a combination of shapers, e.g., TAS for streams that require low latencies and Multi-CQF for the other streams is the best combination in practice. Using TAS for only the streams with tight timing constraints will reduce the search space for GCLs, improving the scalability of the configuration solutions.

The CQF, CSQF and Multi-CQF results have been validated using a simulator developed in Python.

VI. CONCLUSION AND FUTURE WORK

In this paper we have evaluated several CQF traffic shapers, such as the original CQF, the CSQF and the Multi-CQF shapers. We have considered industrial systems where end stations are interconnected via physical links and switches and use IEEE 802.1 TSN.

We have formulated the network configuration problem for these CQF variants and we have developed several solutions, based on CP and SA. The CP formulation can find the optimal solution but does not scale. The SA metaheuristic is able to find good quality solutions in a reasonable time even for large test cases.

We have evaluated CQF, CSQF and Multi-CQF and compared it to TAS. We have used randomly generated test cases following the IEC/IEEE 60802 profile, which were purposely created such that the timing constraints are very tight, to investigate the limits of CQF shapers.

The conclusion is that Multi-CQF can handle well streams even with tight timing constraints. However, as the network size increases, all the peristaltic shapers will lead to larger latencies because the latency of a stream depends on the number of hops and cycle length. In such situations, TAS can be used for the most demanding streams, in combination with CQF shapers.

CSQF, which uses three queues, outperforms CQF, which uses only two queues. They are both outperformed by Multi-CQF if the switch configuration is adequately selected. Determining the switch configuration is an optimization problem, i.e., deciding the priority groups, their number of queues in a group, cycle lengths and bandwidth allocations. However, this optimization problem has not been addressed in this paper and it is left for future work.

In our work we have considered the TAS and CQF shapers in isolation. However, as [33] shows, there are advantages to use combinations of shapers. For example, it is very interesting to combine TAS and CQF in the same switch, such that TAS handles streams with low latency requirements and CQF handles streams that do not have very tight constraints. In our future work we will (i) compare CQF with other shapers besides TAS, e.g., ATS and CBS, and (ii) consider the combination of TAS and CQF, which requires a joint optimization of TAS and CQF configuration, which is different and interesting optimization problem.

REFERENCES

[1] P. Gaj, J. Jasperneite, and M. Felser, "Computer communication within industrial distributed environment—A survey," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 182–189, Feb. 2013.
[2] *IEEE 802.3 Standard for Ethernet*, Standard 802.3, May 2015.
[3] J. D. Decotignie, "Ethernet-based real-time and industrial communications," *Proc. IEEE*, vol. 93, no. 6, pp. 1102–1117, Jun. 2005.
[4] D. Jansen and H. Buttner, "Real-time Ethernet: The ethercat solution," *Comput. Control Eng.*, vol. 15, no. 1, pp. 16–21, Feb. 2004.
[5] J. Feld, "PROFINET—Scalable factory communication for all applications," in *Proc. IEEE Int. Workshop Factory Commun. Syst.*, Aug. 2004, pp. 33–38.
[6] *Aircraft Data Network, Part 7, Avionics Full-Duplex Switched Ethernet Network*, Standard ARINC 664P7, Airlines Electronic Engineering Committee, 2009.
[7] *Time-Triggered Ethernet*, Standard SAE AS6802, SAE International, 2011.
[8] (2016). *Official Website of the 802.1 Time-Sensitive Networking Task Group*. [Online]. Available: http://www.ieee802.org/1/pages/tsn.html
[9] E. Dahlman, G. Mildh, S. Parkvall, J. Peisa, J. Sachs, Y. Selén, and J. Sköld, "5G wireless access: Requirements and realization," *IEEE Commun. Mag.*, vol. 52, no. 12, pp. 42–47, May 2014.
[10] N. Nikaein, C.-Y. Chang, and K. Alexandris, "Mosaic5G: Agile and flexible service platforms for 5G research," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 48, no. 3, pp. 29–34, Sep. 2018.
[11] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. ElBakoury, "Ultra-low latency (ULL) networks: The IEEE TSN and IETF DetNet standards and related 5G ULL research," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 88–145, 1st Quart., 2018.
[12] W. Mahnke, S.-H. Leitner, and M. Damm, *OPC Unified Architecture*. Heidelberg, Germany: Springer, 2009.
[13] O. Kleineberg and A. Schneider, *Time-Sensitive Networking for Dummies, Belden/Hirschmann Special Edition*. Hoboken, NJ, USA: Wiley, 2018.
[14] (2011). *802.1BA—Audio Video Bridging (AVB) Systems*. [Online]. Available: https://standards.ieee.org/ieee/802.1BA/4396/
[15] L. Zhao, P. Pop, Z. Zheng, and Q. Li, "Timing analysis of AVB traffic in TSN networks using network calculus," in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS)*, Apr. 2018, pp. 25–36.
[16] N. Finn. (Oct. 26, 2021). *Multiple Cyclic Queuing and Forwarding*. [Online]. Available: https://www.ieee802.org/1/files/public/docs2021/new-finn-multiple-CQF-0921-v02.pdf
[17] A. Nasrallah, V. Balasubramanian, A. S. Thyagaturu, M. Reisslein, and H. ElBakoury, "Cyclic queuing and forwarding for large scale deterministic networks: A survey," 2019, *arxiv:1905.08478*.
[18] M. Chen, X. Geng, and Z. Li, "Segment routing (SR) based bounded latency," Internet Eng. Task Force, May 2019. [Online]. Available: https://datatracker.ietf.org/doc/draft-chen-detnet-sr-based-bounded-latency/01/
[19] Y. Huang, S. Wang, T. Huang, and Y. Liu, "Cycle-based time-sensitive and deterministic networks: Architecture, challenges, and open issues," *IEEE Commun. Mag.*, vol. 60, no. 6, pp. 81–87, Jun. 2022.
[20] V. Gavriluţ, A. Pruski, and M. S. Berger, "Constructive or optimized: An overview of strategies to design networks for time-critical applications," *ACM Comput. Surv.*, vol. 55, no. 3, pp. 1–35, Apr. 2023.
[21] Y. Seol, D. Hyeon, J. Min, M. Kim, and J. Paek, "Timely survey of time-sensitive networking: Past and future directions," *IEEE Access*, vol. 9, pp. 142506–142527, 2021.
[22] S. M. Laursen, P. Pop, and W. Steiner, "Routing optimization of AVB streams in TSN networks," *SIGBED Rev.*, vol. 13, no. 4, pp. 43–48, Nov. 2016.
[23] N. G. Nayak, F. Dürr, and K. Rothermel, "Routing algorithms for IEEE802.1Qbv networks," *ACM SIGBED Rev.*, vol. 15, no. 3, pp. 13–18, Jun. 2018.
[24] M. A. Ojewale and P. M. Yomsi, "Routing heuristics for load-balanced transmission in TSN-based networks," *ACM SIGBED Rev.*, vol. 16, no. 4, pp. 20–25, Jan. 2020.
[25] S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner, "Scheduling real-time communication in IEEE 802.1Qbv time sensitive networks," in *Proc. 24th Int. Conf. Real-Time Netw. Syst. (RTNS)*, 2016, pp. 183–192.
[26] P. Pop, M. L. Raagaard, S. S. Craciunas, and W. Steiner, "Design optimisation of cyber-physical distributed systems using IEEE time-sensitive networks," *IET Cyber-Phys. Syst., Theory Appl.*, vol. 1, no. 1, pp. 86–94, Dec. 2016.
[27] F. Dürr and N. G. Nayak, "No-wait packet scheduling for IEEE time-sensitive networks (TSN)," in *Proc. 24th Int. Conf. Real-Time Netw. Syst. (RTNS)*, May 2016, pp. 203–212.
[28] E. Schweissguth, P. Danielis, D. Timmermann, H. Parzyjegla, and G. Mühl, "ILP-based joint routing and scheduling for time-triggered networks," in *Proc. 25th Int. Conf. Real-Time Netw. Syst.*, Oct. 2017, pp. 8–17.
[29] M. Pahlevan, N. Tabassam, and R. Obermaisser, "Heuristic list scheduler for time triggered traffic in time sensitive networks," *ACM SIGBED Rev.*, vol. 16, no. 1, pp. 1–6, Feb. 2018.
[30] V. Gavriluţ, L. Zhao, M. L. Raagaard, and P. Pop, "AVB-aware routing and scheduling of time-triggered traffic for TSN," *IEEE Access*, vol. 6, pp. 75229–75243, 2018.
[31] J. Krolikowski, S. Martin, P. Medagliani, J. Leguay, S. Chen, X. Chang, and X. Geng, "Joint routing and scheduling for large-scale deterministic IP networks," *Comput. Commun.*, vol. 165, pp. 33–42, Jan. 2021.
[32] J. Yan, W. Quan, X. Jiang, and Z. Sun, "Injection time planning: Making CQF practical in time-sensitive networking," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Jul. 2020, pp. 616–625.
[33] L. Zhao, P. Pop, and S. Steinhorst, "Quantitative performance comparison of various traffic shapers in time-sensitive networking," *IEEE Trans. Netw. Serv. Manag.*, early access, Jun. 3, 2022, doi: 10.1109/TNSM.2022.3180160.
[34] (2020). *IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 34: Asynchronous Traffic Shaping*. [Online]. Available: https://standards.ieee.org/ieee/802.1Qcr/7420/
[35] E. K. Burke and G. Kendall, *Search Methodologies*. New York, NY, USA: Springer, 2005.
[36] K. Apt, *Principles of Constraint Programming*. Cambridge, U.K.: Cambridge Univ. Press, 2003.
[37] J. Y. Yen, "An algorithm for finding shortest routes from all source nodes to a given destination in general networks," *Quart. Appl. Math.*, vol. 27, no. 4, pp. 526–530, 1970.
[38] H. Mausser, "Normalization and other topics in multi-objective optimization," in *Proc. Fields-MITACS Ind. Problems Workshop*, Sep. 2006, p. 89.
[39] *Google*. Accessed: Mar. 22, 2022. [Online]. Available: https://developers.google.com/optimization
[40] F. Hutter, H. H. Hoos, L. B. Kevin, and T. Stützle, "ParamILS: An automatic algorithm configuration framework," *J. Artif. Intell. Res.*, vol. 36, no. 1, pp. 267–306, May 2009.
[41] M. Barzegaran and P. Pop, "Communication scheduling for control performance in TSN-based fog computing platforms," *IEEE Access*, vol. 9, pp. 50782–50797, 2021.

KONSTANTINOS ALEXANDRIS received the Diploma and M.Sc. degrees (Hons.) in electronic and computer engineering from the Technical University of Crete (TUC), Greece, in 2012 and 2014, respectively, and the Ph.D. degree in communications and electronics from Télécom Paris, France, in 2018. In parallel with his M.Sc. degree, he joined the Telecommunications Circuits Laboratory (TCL), EPFL, Switzerland. From 2014 to 2019, he worked as a Research and Development Engineer from EURECOM in the field of 5G networks. Since 2019, he has been a Senior Research and Development Engineer at Huawei Technologies Duesseldorf GmbH, where he has been working on projects for time-sensitive/deterministic networking and data centers. Specifically, he was involved in several H2020/FP7 collaborative research projects. He contributed to the development of the Mosaic5G initiative and the OpenAirInterface 5G platform features. His research interests include Industry 4.0, Future Internet, and 5G networks. He was a recipient of the 2012–2013 IEEE VTS/AESS Joint Greece Chapter Best Diploma Thesis Award.

PAUL POP (Member, IEEE) received the Ph.D. degree in computer systems from Linköping University, in 2003. He is currently a Professor of cyber-physical systems with the DTU Compute, Technical University of Denmark (DTU). He is also a Coordinator of the Nordic University Hub on Industrial the Internet of Things. He has coordinated the European Training Network on fog computing for robotics and industrial automation. His research interest includes developing methods and tools for the analysis and optimization of networked dependable cyber-physical systems. In this area, he has published over 150 peer-reviewed papers, three books, and seven book chapters. He has served as a Technical Program Committee Member for several conferences, such as DATE and ESWEEK. He received the Best Paper Award from DATE 2005, RTIS 2007, CASES 2009, MECO 2013, DSD 2016, and ETFA 2020, and the Outstanding Paper Award from RTNS 2022.

TONGTONG WANG received the B.S. degree in computer science from the Beijing University of Posts and Telecommunications and the M.S. degree in electrical engineering from Linköping University. She is currently an IP Network Expert at the Wired Network Research Department, Huawei Technologies Company Ltd. After seven years in IEEE 802 Ethernet Standard Research and Development, her current research interest includes network latency guarantee and optimization. She is an Editor of IEEE P802.1DF TSN profile for service provider networks.