Quality-Of-Control-Aware Scheduling of Communication in TSN-Based Fog Computing Platforms Using Constraint Programming

Mohammadreza Barzegaran[1]
DTU Compute, Technical University of Denmark, Kongens Lyngby, Denmark
mohba@dtu.dk

Bahram Zarrin
DTU Compute, Technical University of Denmark, Kongens Lyngby, Denmark
baza@dtu.dk

Paul Pop
DTU Compute, Technical University of Denmark, Kongens Lyngby, Denmark
paupo@dtu.dk

Abstract

In this paper we are interested in real-time control applications that are implemented using Fog Computing Platforms consisting of interconnected heterogeneous Fog Nodes (FNs). Similar to previous research and ongoing standardization efforts, we assume that the communication between FNs is achieved via IEEE 802.1 Time Sensitive Networking (TSN). We model the control applications as a set of real-time streams, and we assume that the messages are transmitted using time-sensitive traffic that is scheduled using the Gate Control Lists (GCLs) in TSN. Given a network topology and a set of control applications, we are interested to synthesize the GCLs for messages such that the quality-of-control of applications is maximized and the deadlines of real-time messages are satisfied. We have proposed a Constraint Programming-based solution to this problem, and evaluated it on several test cases.

2012 ACM Subject Classification Networks → Traffic engineering algorithms; Computer systems organization → Embedded software; Theory of computation → Constraint and logic programming

Keywords and phrases TSN, Fog Computing, Constraint Programming, Quality of Control

Digital Object Identifier 10.4230/OASIcs.Fog-IoT.2020.3

Funding *Mohammadreza Barzegaran*: The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 764785, FORA – Fog Computing for Robotics and Industrial Automation.

1 Introduction

In this paper we focus on Fog Computing Platforms (FCPs) for Industrial Control Applications, consisting of heterogeneous *fog node*s (FNs). We consider that FNs are interconnected using a deterministic communication solutions such as IEEE 802.1 Time Sensitive Networking (TSN) [7]. TSN consists of a set of amendments to the IEEE 802.1 Ethernet standard that introduce real-time and safety critical aspects, e.g., IEEE 802.1Qbv defines a Time-Aware Shaper (TAS) mechanism that enables the scheduling of messages based on a global schedule table. The scheduling relies on a clock synchronization mechanism 802.1ASrev [9], which defines a global notion of time. The configuration of the communication infrastructure in an

[1] corresponding author

FCP has an impact on the performance of controllers, which are the main components of industrial applications. The main focus of this paper is to configure an FCP in terms of the scheduling of messages on TSN [5] such that the control performance is maximized.

Fog Computing has received a lot of attention recently [11], and several researchers have proposed the use of TSN in an FCP as a means of achieving deterministic communication for dependable industrial applications [12]. There has been a lot of work on task scheduling for control performance [14], including considering Fog-based implementations [1]. Although researchers have proposed approaches to derive the schedule tables in TSN for Time-Sensitive (TS) traffic, e.g., via Satisfiability Modulo Theories (SMT) [5] and metaheuristics [13] only one work so far has addressed the issue of control performance [10] for industrial applications. [10] focuses on the problem of routing and scheduling of messages to achieve control stability, but ignores the specifics of scheduling TS traffic in TSN, which does not allow the control of individual frames. Instead, only the status of the queue gates can be controlled via Gate Control Lists (GCLs). This may lead to non-determinism of message scheduling, which has to be carefully considered during the GCL synthesis.

In this paper, we propose a constraint programming (CP)-based GCL synthesis strategy aiming at maximizing the quality-of-control (QoC) . We employ meta-heuristic search strategies in CP solvers to reduce the computation time needed to find optimized solutions.

2 System Model

2.1 Architecture Model

We model the system architecture as a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ and a set of routes $\mathcal{R}$, where $\mathcal{V}$ is a set of vertices that represents nodes, and $\mathcal{E}$ is a set of edges, where an edge represents a physical link between two nodes. A node $\nu_i \in \mathcal{V}$ is either an end-system, which may be the source (talker) or the destination (listener) of a stream, or a switch, which forwards messages to the other nodes. A physical link is a full-duplex bidirectional link $\epsilon_{i,j} \in \mathcal{E}$ (equivalent to $\epsilon_{j,i}$) that logically links the nodes ν_i and ν_j. A logical link $\epsilon_{i,j}$ is characterized by the tuple $< s, d, mt >$ denoting the speed of the port in Mbit/s, the transmission delay of the port and the time granularity (macrotick) of an event for the port in micro-seconds. According to IEEE 802.1Qbv [8], we assume 8 queues for each link $\epsilon_{i,j}$ which connects the egress port of the node ν_i to the ingress port of the node ν_j.

The transmission delay of a link, $\epsilon_{i,j}.d$, is captured by the function $\mathbf{h}(c)$ which gets the size of a stream's frame c, as input, and is given for every link. Given that each stream has a known size and it is forwarded through a port with a known speed, the transmission time of the stream's frames can be easily determined. For example, transmitting a maximum transmission unit (MTU)-sized IEEE 802.1Q Ethernet frame of 1,542 bytes on a 1 Gbit/s link would take 12.33 μs. The MTU-sized frame is the maximum size of a single data unit that can be transmitted over a network.

A route $r_i \in \mathcal{R}$ is an ordered list of links, starting with a link originating in a talker end system, and ending with a link in a listener end system. The number of links in the route r_i is denoted with $|r_i|$. We define the function $\mathbf{u} : \mathcal{R} \times \mathbb{N}_0 \rightarrow \mathcal{E}$ to capture the jth link of the route r_i. We assume that each stream is associated to only one route but several streams may share the same route. We also assume that the streams are unicast which impose that there is only one talker and one listener for a stream. Our model can be extended to multicast streams.

2.2 Application Model

We model a control application as a set of streams $\mathcal{S}$. A stream $s_i \in \mathcal{S}$ is captured by the tuple $< p, c, t, d, j >$ denoting the priority, the message size in bytes, the period in milliseconds, the deadline, i.e., the maximum allowed end-to-end delay, and the maximum allowed jitter, both in milliseconds. Since, we assume 8 queues for each link, the priority of a massage is given from 0 to 7. The number of instances for stream s_i is denoted with $|s_i|$, and is derived from the period of the stream t and the hyperperiod which is the least common multiple of the periods of all streams. For example, for three streams with the periods of 4, 5 and 3 ms, the hyperperiod would be 60 ms and the streams will have 15, 12 and 20 instances respectively.

The stream s_i is transmitted via a route r_j which is captured by the function $\mathbf{z} : \mathcal{S} \rightarrow \mathcal{R}$ that maps the streams to the routings. We define a frame for each instance $0 \leq k < |s_i|$ of the stream s_i and on each link $0 \leq m < |r_j|$ of the route r_j, and denote it with $f^k_{i,m}$. A frame $f^k_{i,m}$ is associated with the tuple $< \phi, l >$ denoting the start time of the frame (offset ϕ) and its duration (length l).

2.3 Time-Sensitive Transmission in TSN

The internal of a TSN switch is depicted in Fig. 1, where the switching fabric receives streams form ingress ports and forwards each stream to the egress port that is determined in the internal routing tables. In this paper, we assume all the streams are using the Time-Sensitive (TS) class for the transmission.

We assume that each of the egress ports has eight priority queues and each priority queue stores the forwarded stream in First-In-First-Out (FIFO) order. A subset of the queues is reserved for Scheduled Traffic (ST) according to the Priority Code Point (PCP) defined in the frame header; and the remaining queues are used for other, less critical, traffic.

According to the 802.1Qbv standard, a gate is associated to each of the queues which controls the traffic flow by opening and closing that are determined in the predefined Gate Control List (GCL). An open gate only allows the transition of queued traffic from the predetermined egress port. When multiple gates are open at the same time on the same egress port, the highest priority queue blocks other gates until closing.

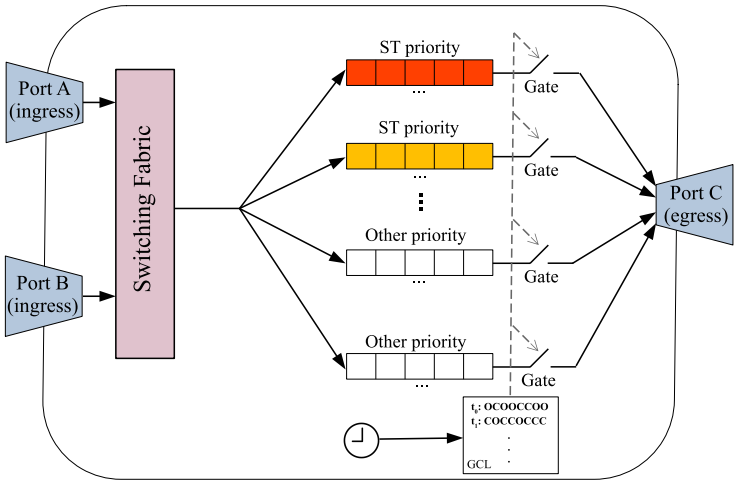

Figure 1 TSN switch internals.

3 Problem Formulation

We formulate the problem as follows: Given (1) a set of streams $\mathcal{S}$, (2) a network graph $\mathcal{G}$, and (3) a set of routings $\mathcal{R}$, we want to determine the GCLs such that the streams are schedulable (their deadlines are satisfied) and the QoC, as defined in Sect. 4, is maximized. In this paper we assume, similar to [5], that the GCLs are deterministic, i.e., the streams are isolated from each other: Only the frames of one of the streams are present in a queue at a time. Hence, the GCL synthesis problem is equivalent to determining (i) the offsets of frames $f^k_{i,m}.\phi$, and (2) their duration $f^k_{i,m}.l$. The offset of a frame maps to when the gate should be open and the duration of the frame maps to how long it should be open.

4 Control Performance

A control application takes input form sensors, processes data, calculates output, and sends the output to actuators. Various communication links are used to link senors and actuators to the processing elements where control output is calculated. A control application is dependent on time, i.e., timing of data sampling from sensors, calculation of control output and actuation of actuators, which affects the control performance. The control performance is degraded when the delay between sampling and actuation is more than what the controller is designed for or when the delay varies in each iteration, see [3] for more details.

In this paper, we consider that the communication between sensors, processing elements and actuators is based on TSN. We schedule the sensor and actuator messages along with other messages, and the control performance degrades when the control-related messages experience jitter, defined in our case as the variation among the end-to-end delays of a message. We use JitterTime to analyse the Quality-of-Control (QoC), which is used interchangeably to mean "control performance". JitterTime simulates a control application using the given timing for sampling and actuation and calculates the QoC using the given quadratic cost function, see [4] for details.

For the examples and test cases in the paper, we consider that the control tasks implement a control application consisting of a dynamical system, and we use a quadratic cost function for JitterTime, similar to [1]. The sensor samples the plant with the same period as the control application and sends a message to the node that runs the control application. The actuator receives a message when the control application produces its output. Fig. 2 shows an example system model.

5 Constraint Programming

Constraint Programming (CP) is a declarative programming paradigm that has been widely used to solve a variety of optimization problems such as scheduling, routing, and resource allocations. 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. The constraint solver systematically performs an exhaustive search by exploring all the possibilities of assigning different values to the variables.

In our future work we will integrate JitterTime with our CP formulation to evaluate *each* visited solution during the search w.r.t. its QoC. However, our approach in this paper is to use the *jitter* as a proxy for the QoC [4]. Hence, we are looking for solutions such that the

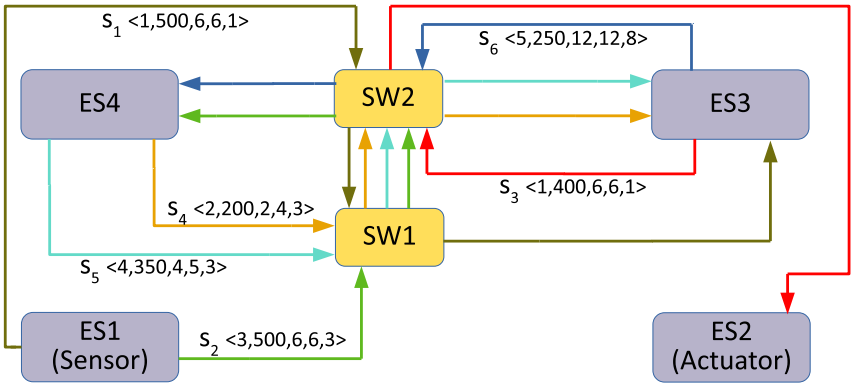

Figure 2 Example system model. One control application with messages of 6 ms period on a FCP-based architecture with a sensor, an actuator, two FNs and two switches. The sensor sends a 500 bytes message to ES3 where control output is being calculated and ES3 sends a message of 400 bytes when output is ready. All links have the speed of 100 Mbps. The routing and parameters of streams are also depicted; coloring distinguishes the different streams.

network and stream constraints defined in Sect. 5.2, are satisfied and the jitter defined in Sect. 5.3 are minimized. We record all the solutions visited that have the *best* cost function, and after the search terminates we use JitterTime to determine their QoC.

In the following sections, we present a CP model for our problem, including the decision variables, constraints, and the objective function. Additionally, we propose different search strategies to improve the search speed.

5.1 CP Model

We define decision variables for the offsets and lengths of frames in our CP model, and we bound them in Eq. (1).

$$\forall s_i \in \mathcal{S}, \forall m \in [0,..,|s_i|), \forall k \in [0,..,|r_j|), r_j = \mathbf{z}(s_i), \epsilon_{v,w} = \mathbf{u}(r_j,k) :$$
$$0 \leq f^k_{i,m}.\phi \leq \frac{s_i.t}{\epsilon_{v,w}.mt} \qquad f^k_{i,m}.l = \frac{s_i.c}{\epsilon_{v,w}.s \times \epsilon_{v,w}.mt} \tag{1}$$

5.2 Constraints

We model the network using five constraints that regulate traffic. A directed physical link transmits one frame at a time, i.e., two frames can not share a physical link at any time, which is modeled with the constraint in Eq. (2):

$$\forall s_i, s_j \in \mathcal{S}, i \neq j, \forall m \in [0,..,|s_i|), \forall n \in [0,..,|s_j|),$$
$$\forall k \in [0,..,|r_o|), r_o = \mathbf{z}(s_i), \forall l \in [0,..,|r_p|), r_p = \mathbf{z}(s_j), \epsilon_{v,w} = \mathbf{u}(r_o,k), \epsilon_{v,w} = \mathbf{u}(r_p,l) :$$
$$(f^k_{i,m}.\phi + m \times \frac{s_i.t}{\epsilon_{v,w}.mt} \geq f^l_{j,n}.\phi + n \times \frac{s_j.t}{\epsilon_{v,w}.mt} + f^l_{j,n}.l) \vee$$
$$(f^l_{j,n}.\phi + n \times \frac{s_j.t}{\epsilon_{v,w}.mt} \geq f^k_{i,m}.\phi + m \times \frac{s_i.t}{\epsilon_{v,w}.mt} + f^k_{i,m}.l). \tag{2}$$

The constraint in Eq. (3) imposes that a stream propagates from the talker to the listener through the ordered links determined in the mapped routing. It also imposes that the frame can only be scheduled to be transmitted after it has completely received by the node

considering the network propagation delay. According to the 802.1AS clock synchronization mechanism [9], the network precision which is the worst-case difference between the nodes clock in the network, is defined and denoted with δ:

$$\forall s_i \in \mathcal{S}, \forall m \in [0,..,|s_i|), \forall k \in [0,..,(|r_j|-1)),$$
$$r_j = \mathbf{z}(s_i), \epsilon_{v,w} = \mathbf{u}(r_j,k), \epsilon_{w,x} = \mathbf{u}(r_j,(k+1)) :$$
$$f^{k+1}_{i,m}.\phi \times \epsilon_{w,x}.mt \geq (f^{k}_{i,m}.\phi + f^{k}_{i,m}.l) \times \epsilon_{v,w}.mt + \epsilon_{v,w}.d + \delta. \tag{3}$$

We also isolate streams in different queues of switches to avoid displacement of frames. The constraint in Eq. (4) imposes that either two frames are not received at the ingress port of a switch at the same time or have different priorities, i.e, one frame is received after or before the other one, or has different priority when they are received at the same time, which enforces their order of transmission in the switch schedule, see [5] for more details:

$$\forall s_i, s_j \in \mathcal{S}, i \neq j, \forall m \in [0,..,|s_i|]), \forall n \in [0,..,|s_j|),$$
$$\forall k \in [1,..,|r_o|), r_o = \mathbf{z}(s_i), \forall l \in [1,..,|r_p|), r_p = \mathbf{z}(s_j),$$
$$\epsilon_{v,w} = \mathbf{u}(r_o,k), \epsilon_{a,b} = \mathbf{u}(r_p,l), \epsilon_{x,v} = \mathbf{u}(r_o,k-1), \epsilon_{y,a} = \mathbf{u}(r_p,l-1) :$$
$$((f^{k}_{i,m}.\phi \times \epsilon_{v,w}.mt + m \times s_i.t + \delta \leq f^{l-1}_{j,n}.\phi \times \epsilon_{y,a}.mt + n \times s_j.t + \epsilon_{y,a}.d) \vee$$
$$(f^{l}_{j,n}.\phi \times \epsilon_{v,w}.mt + n \times s_j.t + \delta \leq f^{k-1}_{i,m}.\phi \times \epsilon_{x,v}.mt + m \times s_i.t + \epsilon_{x,v}.d)) \vee (s_i.p \neq s_j.p). \tag{4}$$

The constraint in Eq. (5) imposes that a stream is received by its listener within its deadline, i.e., the time interval between the scheduled transmission of a stream from its talker and the reception of it by the listener is smaller than its deadline:

$$\forall s_i \in \mathcal{S}, \forall m \in [0,..,|s_i|), r_j = \mathbf{z}(s_i), \epsilon_{a,b} = \mathbf{u}(r_j,0), \epsilon_{y,z} = \mathbf{u}(r_j,(|r_j|-1)) :$$
$$f^{0}_{i,m}.\phi \times \epsilon_{a,b}.mt + s_i.d \geq \epsilon_{y,z}.mt \times (f^{(|r_j|-1)}_{i,m}.\phi + f^{(|r_j|-1)}_{i,m}.l). \tag{5}$$

We also define the constraints for the talkers and listeners in Eq. (6), which imposes that the jitter of every instance of a stream should be within the defined value, which is denoted with $s_i.j$ for the stream s_i.

$$\forall s_i \in \mathcal{S}, \forall m,n \in [0,..,|s_i|), r_j = \mathbf{z}(s_i), \epsilon_{a,b} = \mathbf{u}(r_j,0), \epsilon_{y,z} = \mathbf{u}(r_j,(|r_j|-1)) :$$
$$|(f^{0}_{i,m}.\phi - f^{0}_{i,n}.\phi) \times \epsilon_{a,b}.mt + (m-n) \times s_i.t| \leq s_i.j$$
$$|(f^{(|r_j|-1)}_{i,m}.\phi - f^{(|r_j|-1)}_{i,n}.\phi) \times \epsilon_{y,z}.mt + (m-n) \times s_i.t| \leq s_i.j. \tag{6}$$

5.3 Objective Function

The CP solver finds the first feasible solution that satisfies the presented constraints and determines the values of the CP model variables. The CP solver optimizes the solution concerning the defined objective function. We define an optimization objective to find a solution which schedules streams such that streams have minimum jitter. Although the constraint in Eq. 6 imposes that the jitter is bounded, we seek for a minimum-jitter solution. The proposed optimization objective function Ω accumulates the sending and receiving jitter for every stream, and defined in Eq. (7):

$$\forall s_i \in \mathcal{S}, \forall m,n \in [0,..,|s_i|), r_j = \mathbf{z}(s_i), \epsilon_{a,b} = \mathbf{u}(r_j,0), \epsilon_{y,z} = \mathbf{u}(r_j,(|r_j|-1)) :$$
$$\Omega = \sum |(f^{0}_{i,m}.\phi - f^{0}_{i,n}.\phi) \times \epsilon_{a,b}.mt + (m-n) \times s_i.t|$$
$$+ |(f^{(|r_j|-1)}_{i,m}.\phi - f^{(|r_j|-1)}_{i,n}.\phi) \times \epsilon_{y,z}.mt + (m-n) \times s_i.t| \tag{7}$$

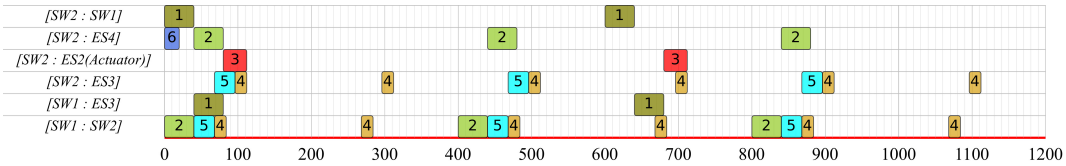

Figure 3 Optimized GCL for the system in Fig. 2. Messages have the same color as the streams in Fig. 2.

5.4 Search Strategies

In this work, we use Google OR-Tools [6] as a CP solver to implement the presented CP model. This CP solver is quite flexible and comes with several extension mechanisms that allow customizing and combining different search strategies such as systematic search, local search, and meta-heuristics algorithms. In this paper, we have used two search strategies; a *Systematic*, and a *Meta-heuristic* strategy.

The first strategy finds the optimal solution by systematically exploring all the possibility of assigning different values to the decision variable. It requires to specify two procedures for the search algorithm. The first is the order of selecting the variables for assignment. The other procedure is the order of selecting the values from the domain of a variable for assignment. Based on our parameter tuning experiments, we choose to use the random order for both procedures (random-variable and random-value).

The second search strategy does not guarantee optimality. Instead, it aims at finding good quality solutions in a reasonable time, and hence it is based on Tabu Search meta-heuristic algorithm [2], which aims to avoid the search process being trapped in a local optimum by increasing diversification and intensification of the search. We have implemented this strategy by extending the OR Tools' implementation of Tabu Search. For intensification, it will keep certain variables bounded to certain values, and for diversification, we will forbid some variables to take some values. We specify two sets of variables for *keep-tenure* and *forbid-tenure*. The variables in the first set must keep their values in the next solution, while the variables in the second set can not use the corresponding values. We also specify the number of iterations or a certain amount of time to keep these variables in these sets.

We run these search strategies for solving the offset variables $f^k_{i,m}.\phi$ as the primary decision variables since they have a direct impact on our cost function. We solve the length variables $f^k_{i,m}.l$ as a constraint satisfaction problem by using *SolveOnce* strategy of the solver which finds the first feasible assignments for these variables.

6 Evaluation

We have evaluated our proposed CP model with several test cases. Our solution is implemented in Java using Google OR-Tools [6] and was run on a computer with an i9 CPU at 3.6 Ghz and 32 GB of RAM, with a time limit of 30 minutes to 5 hours, depending on the size of the test case.

Let us consider the test case in Fig. 2. We schedule the traffic using the *Systematic* and *Meta-heuristic* strategies from Sect. 5.4. Both strategies found the same best solution depicted in Fig. 3 as Gantt chart, which in this case has zero jitter and all streams are schedulable. JitterTime reports a QoC value of 1214 for both solutions. We also measured the run-times of each search strategy, which are 3.67 s for the *Systematic* strategy and 162 ms for the *Meta-heuristic* strategy.

We have also evaluated our solution on progressively larger test cases. The results are presented in Table 1, which shows the 5 additional test cases; Test case 5 is a realistic automotive test case which consists of a TSN-based "fog nodes on wheels" implementation of autonomous driving functions. In the table, the topology of the network is summarized in columns 3 and 4, where we have the number of end-systems and switches, respectively. The values in the column 5 are the maximum jitter for all streams. The values in columns 6 and 7 are the run-time of the solution for respectively *Systematic* and *Meta-heuristic* strategies. As we can see, our CP-based approach is able to find schedulable solutions with zero jitter in all cases. In addition, the *Meta-heuristic* solution scales well with the problem size, and has been able to find the same the optimal solutions as the ones found by the *Systematic* search, in a much shorter time.

However, the improvement in run-time depends on the test case: the search strategy has a big impact on run-time of the solver and the proposed *Meta-heuristic* strategy improves run-time of the addressed scheduling problem.

Table 1 Evaluation results for five test cases.

#	No. of Streams	No. of ESs	No. of SWs	Max. Jitter	Run-Time for Systematic	Run-time for Meta-heuristic
1	10	5	5	0	14:12 min	15.89 s
2	8	5	2	0	2:23 min	3.59 s
3	20	15	15	0	24:44 min	32.2 s
4	20	15	15	0	26:56 min	39.9 s
5	27	20	20	0	42:43 min	2:41 min

7 Conclusions and Future Work

In this paper, we have addressed the problem of real-time communication scheduling on TSN networks on an FCP, aiming at improving the control performance. We have used the scheduled traffic class, which sends the messages based on Gate Control Lists. We have proposed a constraint programming-based solution, modeling the problem constraints as well as objective function for optimizing the network for control applications. The search uses jitter as a "proxy" objective function for the control performance, which has been determined using JitterTime for the best solutions found by the Google OR-Tools solver. As the results show, employing a metaheuristic search in the solver, we can obtain good quality solutions in a short time.

In our future work, we plan to (i) integrate JitterTime into the search process of the CP solver, (ii) integrate task scheduling and message scheduling into a joint QoC-aware CP formulation, and (iii) evaluate the CP approach on larger test cases.

References

1 M. Barzegran, A. Cervin, and P. Pop. Towards quality-of-control-aware scheduling of industrial applications on fog computing platforms. In *Workshop on Fog Computing and the IoT*, 2019.
2 Edmund K Burke, Graham Kendall, et al. *Search methodologies*. Springer, 2005.
3 A. Cervin, D. Henriksson, B. Lincoln, J. Eker, and K.-E. Årzén. How does control timing affect performance? analysis and simulation of timing using Jitterbug and TrueTime. *IEEE Control Systems Magazine*, 23(3):16–30, June 2003.

4 A. Cervin, P. Pazzaglia, M. Barzegaran, and R. Mahfouzi. Using JitterTime to analyze transient performance in adaptive and reconfigurable control systems. In *IEEE International Conference on Emerging Technologies and Factory Automation*, pages 1025–1032, 2019.

5 Silviu S Craciunas, Ramon Serna Oliver, Martin Chmelík, and Wilfried Steiner. Scheduling real-time communication in ieee 802.1 qbv time sensitive networks. In *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, pages 183–192, 2016.

6 Google. Google OR-Tools. `https://developers.google.com/optimization`, Accessed on Jan 2020.

7 IEEE. *Official Website of the 802.1 Time-Sensitive Networking Task Group*, 2016 (accessed December. 12, 2018). URL: `http://www.ieee802.org/1/pages/tsn.html`.

8 IEEE. 802.1Qbv—enhancments for scheduled traffic. `https://www,ieee802.org/1/pages/802.1bv.html`, 2016 Draft 3.1.

9 IEEE. 802.1ASrev—timing and synchronization for time-sensitive applications. `http://www.ieee802.org/1/pages/802.1AS-rev.html`, 2017.

10 Rouhollah Mahfouzi, Amir Aminifar, Soheil Samii, Ahmed Rezine, Petru Eles, and Zebo Peng. Stability-aware integrated routing and scheduling for control applications in Ethernet networks. In *Design, Automation & Test in Europe Conference*, pages 682–687, 2018.

11 Carla Mouradian, Diala Naboulsi, Sami Yangui, Roch H. Glitho, Monique J. Morrow, and Paul A. Polakos. A Comprehensive Survey on Fog Computing: State-of-the-Art and Research Challenges. *IEEE Communications Surveys and Tutorials*, 20(1):416–464, 2018.

12 P. Pop, M. L. Raagaard, M. Gutierrez, and W. Steiner. Enabling fog computing for industrial automation through Time-Sensitive Networking (TSN). *IEEE Communications Standards Magazine*, 2(2):55–61, 2018.

13 Paul Pop, Michael Lander Raagaard, Silviu S Craciunas, and Wilfried Steiner. Design optimisation of cyber-physical distributed systems using IEEE Time-Sensitive Networks. *IET Cyber-Physical Systems: Theory & Applications*, 1(1):86–94, 2016.

14 Zhi Wen Wang and Hong Tao Sun. Control and scheduling co-design of networked control system: Overview and directions. In *in Proceedings International Conference on Machine Learning and Cybernetics*, volume 3, pages 816–824, 2012.