Combining Static and Dynamic Traffic with Delay Guarantees in Time-Sensitive Networking

Lisa Maile(✉), Kai-Steffen Hielscher, and Reinhard German

Computer Networks and Communication Systems, Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany
{lisa.maile,kai-steffen.hielscher,reinhard.german}@fau.de

Abstract. To support reliable and low-latency communication, Time-Sensitive Networking introduced protocols and interfaces for resource allocation in Ethernet. However, the implementation of these allocation algorithms has not yet been covered by the standards. Our work focuses on deadline-guaranteeing resource allocation for networks with static and dynamic traffic. To achieve this, we combine offline network optimization heuristics with online admission control and, thus, allow for new flow registrations while the network is running. We demonstrate our solution on Credit-Based Shaper networks by using the delay analysis framework Network Calculus. We compare our approach with an intuitive and a brute-force algorithm, where we can achieve significant improvements, both, in terms of quality and runtime. Thereby, our results show that we can guarantee maximum end-to-end delays and also increase the flexibility of the network while requiring only minimal user input.

Keywords: Performance modeling · Network optimization · Latency guarantees · Auto-configuration · Resource allocation · Time-sensitive networking

1 Introduction

With the increasing need for ultra-reliable and low latency communication, there has been a growing demand for more advanced transmission mediums to support them. As a response, the IEEE Time-Sensitive Networking (TSN) task group developed new standards that, i.a., allow for resource allocation in Ethernet with the introduction of central configuration units and decentralized reservation protocols. However, the current standards do not provide guidance on the configuration of TSN networks. Therefore, several works have addressed this by proposing either offline (e.g., [6,10]) or online (e.g., [11,18]) configuration approaches. However, offline solutions do not allow for variable traffic and dynamic flows during the runtime of the network, while online solutions require the a-priori definition

E. Kalyvianaki and M. Paolieri (Eds.): VALUETOOLS 2023, LNICST 539, pp. 117–132, 2024.
https://doi.org/10.1007/978-3-031-48885-6_8

of non-trivial delay bounds by a user. To this end, we introduce a novel tool that combines offline and online configuration to support static and dynamic traffic while providing delay guarantees for time-sensitive flows. Thereby, offline configuration optimizes network setup in advance, while online configuration enables flow reservation and de-reservation while the network is running.

Combining both offline and online resource reservation with delay constraints is crucial in the design and operation of Industry 4.0 networking systems. Firstly, offline optimization allows for the creation of an optimal plan for resource allocation and scheduling. This ensures that the network is designed to operate at peak efficiency and that resources are allocated effectively. However, network conditions are rarely static and can change over time due to varying traffic patterns, new applications, devices, or other factors. Online admission control allows the network to respond quickly to changing conditions while maintaining its performance.

In this paper, we present a new approach that combines offline and online resource reservation for time-sensitive communication. The goal is to offer an auto-configuration framework which eliminates the need for manual intervention while still offering reliable flow delays. To the best of our knowledge, we are the first to propose a combined solution for offline and online configuration for TSN with safe delay guarantees.

The remainder of this paper is structured as follows. Section 2 introduces related work and motivates the problem. Section 3 presents an overview of our framework and its relation to the TSN standards. We explain our heuristic optimization in Sect. 4. Afterwards, we extensively evaluate our approach in Sect. 5. Finally, Sect. 6 concludes this paper.

2 Problem Definition and Related Work

Traditional deadline-constrained optimization approaches take a set of flows as input and determine the optimal configuration of network resources, such as bandwidth or time-slots, as output. For example, when using time-triggered gates, individual time-slots are assigned for each flow, e.g., [4,6]. For other schedulers, such as the Credit-Based Shaper (CBS), optimization approaches optimize the reserved bandwidth per traffic class, called idleSlope, so that all flows keep their deadline constraints, e.g., in [10,16]. However, all of these approaches have in common that, when additional time-sensitive flows are added to the network later, the reserved resources, such as time-slots or bandwidth, need to be adapted. This affects existing flow reservations, which evokes the necessity to re-validate all reservations, making these approaches inapplicable for dynamic networks.

To avoid this, flexible online solutions have been proposed in the last years, such as [11,13,18,19]. They offer the opportunity to add and remove flow reservations while the network is running and still provide safe deadline guarantees. The underlying idea is that, instead of configuring resources such as time-slots or bandwidth, each hop is configured with priority-dependent maximum delay

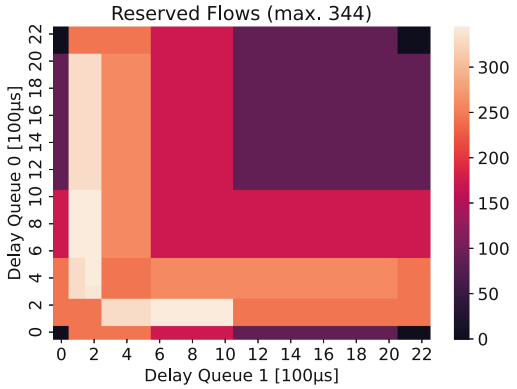

Fig. 1. Impact of delay bounds for individual priority queues on the number of successful flow reservations.

bounds. When the network changes, each hop is allowed to change its configuration, e.g., adapting the reserved bandwidth. Thereby, for each hop, it is validated that the new configuration does not violate the pre-configured delay bounds. This validation can either be done in a central controller instance, such as in [13,18], or with a decentralized reservation protocol, as proposed by [11,19].

While online solutions offer high flexibility, we showed in [18] that the choice of pre-configured delay bounds highly influences the possibility of reserving flows. We illustrate this in Fig. 1, where we used the approach in [18] to reserve flows. We assume a single-hop with two priority queues and a maximum of 344 flows[1] for this example. Figure 1 varies the delay bounds for the two priority queues. We can observe that their choice is of uttermost importance and non-trivial. High delay bounds would lead to a violation of the flows' deadlines, whereas low delay bounds require the reservation of more resources (e.g., more bandwidth to ensure a faster transmission) and, thus, allow for fewer flow reservations in total. Thus, the optimal delay bound values have to be chosen carefully and must neither be too strict nor too loose. Deriving the best delay bounds gets even more complex when more than two priority queues and/or more hops have to be configured.

Grigorjew et al. [12] highlight the fact that brute-forcing the optimal values does not scale for industrial use-cases. Therefore, they employ different machine-learning techniques and show that they can determine fitting delay bounds in the network. However, they assume that the flows' characteristics at each hop are known in advance, which removes the flexibility of online admission control schemes. Besides, in their evaluation, each hop in the network is configured identically, which may lead to potentially sub-optimal results.

[1] We defined the flows according to the traffic profiles 1, 2, 3, and 4 as defined in Table 1, with 86 flows per profile and a bandwidth of 1 Gbit/s.

Table 1. Traffic profiles for industrial scenarios [12]

	Sending Interval	Max. Frame Size	Max. Latency
Profile 1	250 µs	64B	250 µs
Profile 2	500 µs	128B	500 µs
Profile 3	1000 µs	256B	1000 µs
Profile 4	2000 µs	512B	2000 µs
Profile 5	4000 µs	1024B	4000 µs

In real applications, a set of static flows is often given, but operators might also wish for some flexibility to adapt the network during runtime. To this end, we propose our combined approach, where an offline optimization derives delay bounds for the online admission control. For our proof-of-concept, we use [18] to demonstrate the applicability of our solution to be used with online admission control. However, it should also be mentioned that this concept can be applied to any admission control scheme which requires delay bounds as input, such as [11,13,18,19]. We use Network Calculus (NC)—as it is a well-established delay analysis framework—to validate the delay bounds when flows are added to the network. NC models complex communication systems to derive performance guarantees, e.g., maximum per hop and end-to-end delays. To achieve this, worst-case maximum arrival and minimum departures of traffic are modeled using cumulative functions and combined using expressions from the min-plus algebra [15]. For an introduction to NC, see [15]. We provide an overview of NC models for TSN in [17].

3 Network Configuration Framework for TSN

We present a framework for the configuration of TSN networks, which provides safe delay guarantees in networks which combine static and dynamic traffic flows. We refer to *offline configuration* as the configuration of bridge parameters, e.g., the number of priority queues and the worst-case per-hop delay bounds. With *online configuration* (or admission control), we refer to the registration and deregistration of flows while the network is running, after checking the flow's path for sufficient resources. Our goal is to offer delay-guaranteeing networks with only minimal required manual input. Therefore, we defined five traffic profiles for flows in Table 1 [12]. The profiles are derived from PROFINET use-cases for industrial sensor-controller networks. We use the term stream and flow interchangeably.

Input: The only input that we require is 1) the network topology, 2) the maximum number of priority queues, and 3) the percentage of bandwidth that shall remain free for future flow reservations. The latter is provided per link and per traffic profile. In addition, our approach can be optionally supplied with a set of static flows.

Offline Network Optimization: We then run a meta-heuristic optimization which determines the number of required priority queues per hop, as well as the

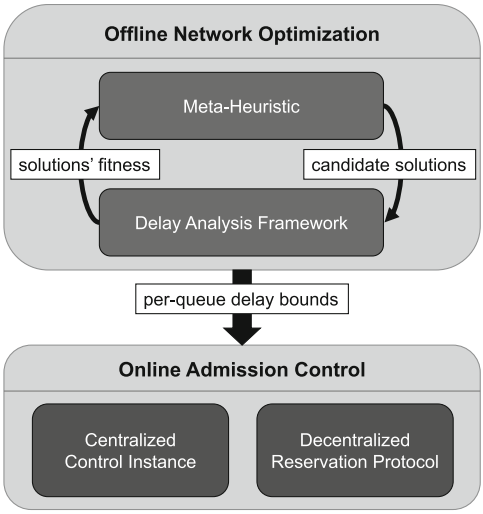

Fig. 2. Framework overview. We use network optimization to derive delay bounds which improve the performance of online admission control.

delay bounds for each priority queue in the network. This is illustrated in Fig. 2. Our heuristic generates potential candidate solutions, which define the delay bounds per priority queue. Each candidate solution is then evaluated according to its potential to provide real-time latency guarantees for the defined traffic profiles. This is expressed as a fitness value, which is defined by the objective function in Sect. 4.2. The latency guarantees are validated using a delay analysis framework. For our tool, we implemented the delay analysis framework using NC, as NC allows for the efficient analysis of large and complex networks. The details of our delay analysis using NC can be found in [18]. The heuristic then iteratively aims at maximizing the fitness value returned by the objective function. We compare two meta-heuristic algorithms, Particle Swarm Optimization (PSO) [14] and Genetic Algorithm (GA) [9], as they have been widely used in various fields.

Online Admission Control: The best candidate solution then defines the delay bounds for each bridge, which are used to configure the network. These bounds remain static, while the actual flow reservations can change adaptively to the network setup, with no re-configuration of the network required. To add new flows to or remove existing flows from the network, existing online admission control algorithms are used, as proposed in [11,13,18,19].

Our framework is suitable for centralized and decentralized network architectures, utilizing protocols specified in the respective standards. In a centralized network, the configuration is done in a so-called Central Network Controller (CNC), as defined by the IEEE 802.1Qcc-2018 standard [1]. The CNC is a separate instance responsible for the reservation and configuration of all flows and nodes in the network. It is aware of the complete network topology, similar to a Software-Defined Networking controller. To communicate with the TSN network

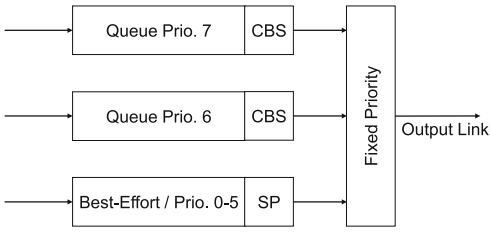

Fig. 3. Example output port with CBS queues.

devices, network management protocols are used, such as the Network Configuration Protocol (NETCONF) defined in RFC 6241.[2] For decentralized networks, the Stream Reservation Protocol (SRP) has been proposed for online admission control in CBS networks. However, as we proved in [19], the delay values proposed for SRP do not cover the worst-case. In 2018, the IEEE TSN group started working on the successor of SRP, the Resource Allocation Protocol (RAP) [2]. Currently, RAP is still undergoing active development and is available in draft version 0.6. RAP considers introducing a pre-configuration phase of the network prior to online admission. This aligns perfectly with the requirements of our framework. Although the existing standards define the necessary protocols, they currently lack a solution for implementing resource allocation with guaranteed delays. Our framework fills this gap, and we hope that our research will help future standardization processes.

4 Meta-Heuristic Optimization

To derive the per-hop delay bounds for the network, we compare two meta-heuristic optimization approaches, PSO and GA. Initially, both algorithms start with a population of random solutions, which we call candidate solutions. The heuristics then successively update their solutions to improve their fitness. Eventually, the heuristics converge to a (near-optimal) solution.

TSN networks allow for up to eight queues in each output port of a bridge. Each queue is configured with a transmission selection algorithm. If multiple queues have eligible packets at the same time, the transmission is defined by the queues' priority. Figure 3 illustrates this for an example output port with two CBS queues. We define a subset of these queues as *priority queues*, meaning that our framework optimizes them for time-critical traffic with deadline guarantees using CBS. All other queues can then be used by non-time-critical traffic without admission control and are not included in our optimization.

4.1 Design Decisions

Our framework includes the following considerations, to ensure that a user only has to provide a minimum of input.

[2] https://datatracker.ietf.org/doc/html/rfc6241.

Routing: We use a k-shortest delay-constrained routing algorithm [22], with the links' bandwidth utilization as weight-function. We chose to weight the bandwidth utilization as this balances the network load, and thus, prevents bottlenecks which would decrease the possibility of online flow reservations.

Flow to Priority Assignment: We do not require any kind of flow-to-priority mapping. Instead, our frameworks assign the priority to each flow automatically, by using the k-shortest paths and checking the priorities on each path with an end-to-end delay that is closest to a flow's deadline successively. If no path allows for a reservation, the fitness of the solution decreases. To increase flexibility, we allow for individual per-hop priorities for each flow, instead of mapping a flow to a single priority on its whole path.

Number of Priority Queues: The required number of priorities in a network is an optimization itself. We only require the definition of a maximum number of priority queues instead. Our heuristic will try to find a schedule with no high-priority queue unused, but potentially without using low priorities. Then, unused queues can be left out or used for other purposes, such as non-time-critical traffic.

Discussion: The above considerations do not have any claim on optimality. E.g., instead of balancing the network, we could add the routing decision as part of the solution, thereby offering it as a variable for the optimization. All of these problems have been covered in heuristics themselves (e.g., [5,8,16,21]), and including them increases the solution space and the runtime significantly. Our results show that with the above decision, already highly practical networks can be built.

Note that we also allow for more detailed modeling, e.g., on the path of flows that arrive during online reservation. The more information is available, the better our approach can configure the network. However, to be appealing for real-life scenarios, we wanted to reduce the minimum required input as much as possible.

4.2 Multi-objective Function

Each candidate solution is evaluated based on its capability of reserving all static flows and on the flexibility it offers for future reservations. The heuristics aim at maximizing the following fitness function, defined for each candidate solution s as

$$f(s) = \omega_1 \cdot f_R(s) + \omega_2 \cdot f_A(s) + \omega_3 \cdot f_D(s), \tag{1}$$

where $\omega_i (i = 1, 2, 3)$ define the weight for each of the objectives. We normalized the fitness value, so f and $f_x (x = \{R, A, D\})$ are between 0 and 1.

Thereby, $f_R(s) = 1$, if all flows from the (optional) set of static offline flows can be reserved with the solution s. $f_R(s)$ can be derived by checking for each static flow whether a path can be found which will meet their end-to-end delay requirement. The ratio of successful reservations then defines $f_R(s)$. Even with a suboptimal solution, the flows' deadlines are still safe. We only reduce the maximum number of reservable flows in the network.

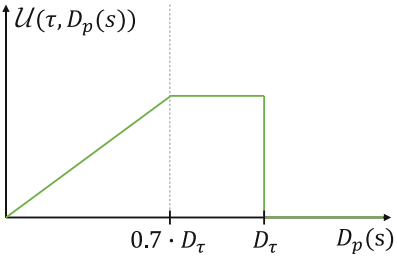

Fig. 4. Utility function for flow τ with delay requirement D_τ and path delay D_p

$f_A(s) = 1$, if it is possible to reserve the required percentage of bandwidth for the online flows. We determine f_A by artificially increasing the arrival function at each link for which the user assigned additional bandwidth for a traffic profile. This is done by dividing the end-to-end delay requirement of each profile into smaller per-hop delays, using the maximum path length of the network. We then check up to which arrival rate we can still guarantee the specified per-hop delay. As we are not aware of the path of the future flows and, thus, do not know the links from which they will arrive at each hop, we use the maximum rate that is possible for each link. f_A then reflects the maximum ratio of available bandwidth when compared to the user's input.

Finally, $f_D(s) = 1$, if the end-to-end delay for the set of offline flows matches their deadline (within a 30% flexibility interval). For each flow τ, we define a utility function $\mathcal{U}(\tau, D_p(s))$ [10] to ensure that its end-to-end delay requirement, D_τ, matches the path delay $D_p(s)$ for the solution s. This is illustrated in Fig. 4. $f_D(s)$ is then defined as

$$f_D(s) = \frac{\sum_{\tau \in \mathcal{F}} \mathcal{U}\big(\tau, D_p(s)\big)}{|\mathcal{F}|}, \tag{2}$$

where $\mathcal{F}$ is the set of static flows. The reason for f_D is that, if flows are scheduled faster than required, more bandwidth than necessary is reserved, which reduces the performance of best-effort traffic. We define a candidate solution to be invalid, if some bridges in the network only utilize their low-priority queues, whereas the high-priority queues cannot be used for time-sensitive traffic. In these solutions, the high-priority queues would be configured unnecessarily. Therefore, these invalid solutions will be rewarded with a fitness value of 0.

We prioritize the individual objectives in our fitness function as follows: We assume that the successful reservation of the static offline flows is of the most importance. Afterwards, we optimize the bandwidth for future reservations of time-sensitive flows. Only if the above two goals are not hindered, the solution might be improved for best-effort and lower priority traffic. As a result, we chose the weight functions to reflect this prioritization as defined in Table 2.

Table 2. Parameter selection for meta-heuristics

	Values
PSO parameters	
w, c_1, c_2	0.5, 2, 2.4
GA parameters	
Crossover operation	blend crossover, $P_c = 0.45, \alpha = 0.15$
Mutation operation	NSGA-II, $P_m = 0.45, \eta = 70, P_{ind} = 0.3$
Selection algorithm	NSGA-II, $P_s = 0.5$
Shared parameters	
ω_1, ω_2, ω_3	0.9, 0.09, 0.01
Population size, convergence	Uniform config.: 100, 15; Individual config.: 200, 20

4.3 Individual Per-Hop Delays

The offline optimizations introduced in Sect. 2, [10,12], configured each hop in the network identically. We evaluate whether individual configurations for each network device can improve the networks' performance. As individual per-hop delays significantly increase the solution space, the heuristics potentially have a higher chance to remain in local optima, instead of finding the global best solution. Therefore, we propose a new approach, where we initialize the solution for individual delays with the results from the uniform network configuration. With this approach, the individual heuristic is guaranteed to perform equally or better than the uniform approach. We will show that this initialization can significantly improve the performance of the algorithms.

4.4 Parameter Choice

We have implemented both, PSO and GA, using Python3. For the GA implementation, we used the evolutionary computation framework DEAP [7]. To determine generic and efficient parameters for both approaches, we used the hyperparameter optimization framework Optuna [3]. To prevent overfitting, we created an evaluation set with highly variable network topologies, different numbers of priorities, and changing flow characteristics. The topologies are presented in Sect. 5.

GAs are inspired by the principles of biological evolution and natural selection. The candidate solutions are updated using selection, crossover, and mutation operators. Selection involves choosing a percentage of P_s individuals from the current population based on their fitness values. Crossover and mutation then adapt a certain percentage of individuals (P_c and P_m, respectively) to explore the solution space. We evaluated all available GA mutation, crossover, and selection algorithms in DEAP, along with their parameter values.

Similarly, PSO iteratively updates the candidate solutions based on the solution's own experience and the collective knowledge of the population. The update uses an inertia w, a cognitive component c_1, and a social component c_2 as parameters. The inertia term allows particles to maintain their momentum, while the cognitive component focuses on the particle's personal best position, and the social component emphasizes the global best position.

The best set of parameters is provided in Table 2. We also use them for our evaluation.

5 Evaluation

We have defined three different topologies which are typical for sensor-controller networks in the industry. Controllers refer to Programmable Logical Controllers (PLCs) and represent end-stations which control industrial machines, e.g., in manufacturing processes. We assume 1 Gbit/s links in all topologies and test cases. We used the following topologies, as they are typical for industrial use-cases: 1) a *line* topology with four bridges, an end-station connected to each of them, and a PLC at the end of the line, 2) a *star-of-stars* topology with one central bridge connected to four bridges, which are again connected to four end-stations, 3) a ring topology with five bridges and each bridge connected to two end-stations. For topologies 2) and 3), we randomly assign one end-device to be the PLC. In all scenarios, flows are addressing the PLC in the network to create a bottleneck for our evaluation. We defined a discrete search space S with steps of 10 µs between 0 and 4 ms, for all algorithms, as 4 ms is the maximum possible deadline for our flows.

5.1 Benchmark Algorithms

We have implemented two approaches as benchmark algorithms for our evaluation. Thereby, we want to evaluate both, the increase in optimality when compared to an intuitive configuration and the gain of performance when compared to an exhaustive search. We evaluate our solution in both categories by using the following two approaches. For comparison, all code is executed without parallelization on an Intel Xeon Silver 4215R processor with 3.20 GHz.

Exhaustive Search (ES): The exhaustive search will iteratively evaluate all possible solutions, and thus, can determine the optimum within the search space. With this approach, we can evaluate how close our results are to the optimal value. However, the exhaustive search suffers from a high runtime. Thus, it cannot cover individual per-hop delays, but will only investigate settings with identical delays for all hops. We will show in our evaluation that, due to this limitation, our solution is able to achieve even better configurations than the exhaustive search.

Intuitive Approach (IA): Our intuitive approach reflects the configuration of a user. It will serve as a benchmark to evaluate whether our heuristics are actually needed to configure high-performance networks. Simply spoken, the intuitive approach will uniformly distribute the end-to-end delays of each flow over its number of hops on the path. The resulting values represent the per-hop delays which each flow requires to meet its deadline requirement. We then configure the network by deriving the quantiles from this set of delays in a way that each flow can be covered by one of the queues. E.g., for four queues, we use the minimum per-hop value, plus the 25, 50, and 75% quantile of the resulting delays to configure the four queues. When compared to our solution, this approach has only minimum runtime, but it does not consider the effect that traffic load has on the queuing delays in the network. As a result, our solution is evaluated against the intuitive approach, i.e., to check for improved fitness values alias more successful flow reservations and evaluate the overhead of performing offline optimization.

5.2 Convergence

Figure 5 illustrates the behavior of the GA and PSO algorithm when compared to the ES in the star-of-stars network, with 150 flows randomly assigned to traffic profiles and end-stations. We assumed a maximum of two CBS queues and a uniform configuration for all network devices. With these settings, the ES can obtain the optimum in 119 min, and we can see how the heuristics perform in comparison. Figure 5 shows the result after each iteration for 30 independent replications, where we consider the algorithms as converged if their results remain constant for 10 iterations. The final result after the algorithms have converged

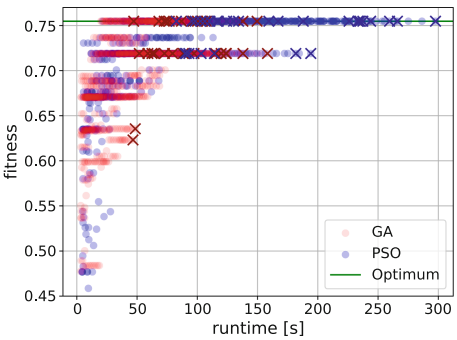

Fig. 5. Results of GA and PSO after each iteration, with results at termination marked as X. The optimum is derived using ES.

is marked with an X. As we can see, both heuristics achieve good results, but GA is faster in reaching high fitness values than PSO. GA performs faster as it only updates solutions with a specific probability (P_c and P_m of Table 2).

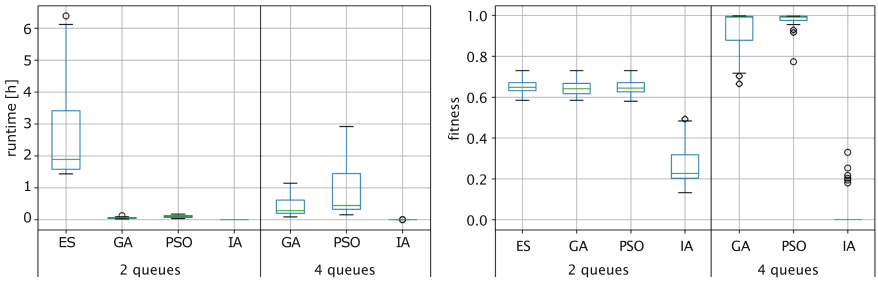

Fig. 6. Runtime and fitness results in networks with uniform bridge configuration.

However, GA is twice as likely to converge in a local rather than global optima when compared to PSO.

5.3 Comparison

To compare our heuristics, we conduct experiments by uniformly choosing one of the three topologies. We randomly generated 200 flows uniformly distributed with the profiles defined in Table 1, originating from random sources and heading to the PLC of the network. We repeat the experiments 30 times for each algorithm. We first assume that all hops are configured with the same delay values.

Figure 6 shows the runtime and resulting fitness values for the experiments, respectively. With the given search space and two priorities, the ES has to evaluate $|S|^2 \approx 160 \cdot 10^3$ possible solutions. Increasing the number of priorities to four would result in $|S|^4 \approx 25 \cdot 10^9$ solutions, which cannot be accomplished by the ES due to its performance.

As we can see in Fig. 6, both heuristics can achieve similar fitness values as the ES for two priorities. While the intuitive approach does allow for some reservations, it cannot compete with the results from our heuristics. For two priorities, the heuristics perform 2.4 times better on average. Additionally, the heuristics can evaluate more priorities than the ES, which results in more successful reservations. E.g., for four priority queues, GA performed 41.4% and PSO 49.3% better than the ES with two priorities. Again, the GA provides better performance in terms of runtime than the PSO, but is more likely to converge in local optima. For the four priority settings, the intuitive approach frequently results in invalid configurations, where high-priority queues remained unused.

5.4 Individual Per-Hop Delays

Individual per-hop delays allow for a better distribution of the network resources when the traffic load is not evenly distributed. For this, we defined the network shown in Fig. 7, based on the PROFINET design guidelines [20] with one priority per flow. Each line is connected to an end-station with an individual traffic

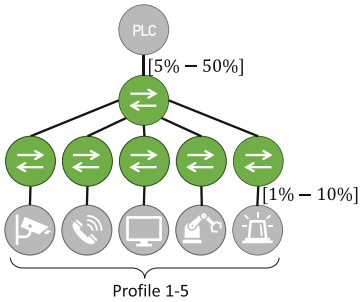

Fig. 7. PROFINET network with given link utlizations.

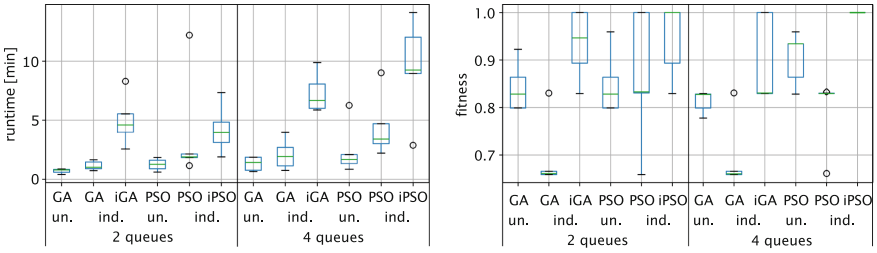

Fig. 8. Runtime and fitness results in a network with individual per-hop configuration.

profile. We randomly generated flows to simulate a link utilization between 5% and 50% on the link to the PLC.

While neither the ES nor the intuitive approach allow for individual per-hop delays, both of our heuristics can evaluate such configurations. Figure 8 shows the result for the GA and PSO algorithms. We compared the uniform network configuration ('un.') with the individual configuration ('ind.'). In addition, we show the results for our initialized solutions ('iGA' and 'iPSO'), which we initialize with the results from the uniform configuration to determine individual delay bounds. We can see that simply allowing for individual queue delays can reduce the performance of the heuristics when compared to the uniform configuration results. This is due to the vast increase of the solution space, which makes it more likely for the heuristics to converge in local optima. However, our initialized algorithms iGA and iPSO can improve the results from the uniform configuration by 10–11% on average and, at the same time, are guaranteed to not provide lower results.

5.5 Flows During Runtime

Finally, to evaluate the benefit for future flow reservations, we again use the star-of-stars topology and define 10 offline flows for each of the profiles 1, 3, and 5. We run different scenarios, where in each scenario, we reserved 50% of the bandwidth for one of the five traffic profiles for future flows. For each sce-

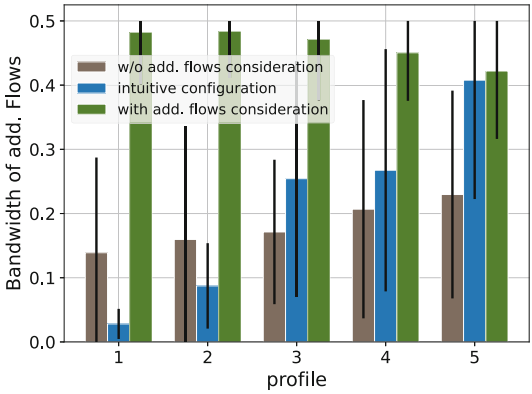

Fig. 9. Successful flow reservations with online admission and one standard deviation. In separate networks, each profile should receive 50% bandwidth.

nario, we repeated the experiment 10 times, resulting in 50 repetitions in total. Figure 9 compares the number of successful reservations during online admission, when the offline optimization considers the bandwidth for the future flows, in comparison to when they are not considered. Additionally, we also compared our heuristics to the intuitive configuration. For these scenarios, we improved the intuitive approach by artificially adding every possible future flow. As we can see, all configurations have some potential to add additional flows during the runtime of the network. However, our combined solution which considers the required bandwidth for future flows can increase the number of successful reservations when compared to the intuitive approach and a configuration without future flow considerations significantly, illustrating our framework's flexibility to support dynamic traffic during runtime. Figure 9 also shows that flows with tight deadline guarantees (e.g., profile 1) profit most from our optimization, while higher latency requirements (e.g. profile 5) are easier to integrate even without specific considerations.

6 Conclusion

We have presented our framework for offline and online configuration of TSN networks with delay guarantees. Thereby, the offline configuration allows for efficient usage of the network resources, while the online configuration allows for the registration and deregistration of flows while the network is running. By considering future flows in the offline configuration phase, we ensure that the network can easily react to changing network conditions while still offering guarantees for the end-to-end delays of time-critical flows. Our framework requires only minimal user input, making it highly relevant for practical application and providing a first step towards the auto-configuration of TSN networks.

We have evaluated two meta-heuristics for our framework and showed that a particle swarm optimization provides better results when compared to a genetic

algorithm, with slightly higher runtimes. We also implemented two benchmark algorithms, one exhaustive search to evaluate the optimum for simple configurations, and one intuitive approach to reflect the behavior of a user. We could show that our heuristics reached similar results as the exhaustive search in just a fraction of the time. Due to the high flexibility and the low runtime, our framework can provide even better results than the exhaustive search and significantly better results than an intuitive solution. We demonstrated that allowing individual delay bounds on each hop improves the network's performance. Finally, we also showed the effect on the success of new flow reservations while the network is running. Thereby, considering the required bandwidth of future flows during the offline configuration highly increases the chance for future flow reservations.

As future work, we want to extend the delay-guaranteeing admission approaches as proposed by [11,13,18,19] to more schedulers. Specifically, we would like to cover solutions for the Asynchronous Traffic Shaper and Cyclic Queuing and Forwarding networks, to ensure that our framework is applicable for a wide range of TSN scenarios.

References

1. IEEE standard for local and metropolitan area networks-bridges and bridged networks—amendment 31: stream reservation protocol (SRP) enhancements and performance improvements. IEEE Std 802.1Qcc-2018 (Amendment to IEEE Std 802.1Q-2018) (2018). https://doi.org/10.1109/IEEESTD.2018.8514112
2. P802.1Qdd—resource allocation protocol (2019). https://1.ieee802.org/TSN/802-1qdd/
3. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: a next-generation hyperparameter optimization framework. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2019)
4. Arestova, A., Hielscher, K.S.J., German, R.: Design of a hybrid genetic algorithm for time-sensitive networking. In: Hermanns, H. (ed.) Measurement, Modelling and Evaluation of Computing Systems, pp. 99–117. Springer International Publishing, Cham (2020)
5. Chuang, C.C., Yu, T.H., Lin, C.W., Pang, A.C., Hsieh, T.J.: Online stream-aware routing for TSN-based industrial control systems. In: 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), vol. 1, pp. 254–261 (2020). https://doi.org/10.1109/ETFA46521.2020.9211969
6. Craciunas, S.S., Oliver, R.S., Chmelík, M., Steiner, W.: Scheduling real-time communication in IEEE 802.1Qbv time sensitive networks. In: Proceedings of the International Conference on Real-Time Networks and Systems, pp. 183–192. ACM Press (2016). https://doi.org/10.1145/2997465.2997470
7. Fortin, F.A., De Rainville, F.M., Gardner, M.A., Parizeau, M., Gagné, C.: DEAP: evolutionary algorithms made easy. J. Machine Learn. Res. **13**, 2171–2175 (2012)
8. Francés, F., Fraboul, C., Grieu, J.: Using Network Calculus to optimize the AFDX network, p. 9 (2006)
9. Fraser, A., Burnell, D.: Computer Models in Genetics. McGraw-Hill, New York (1970)

10. Gavriluţ, V., Pop, P.: Traffic-type Assignment for TSN-based Mixed-criticality cyber-physical systems. ACM Trans. Cyber-Phys. Syst. **4**(2), 1–27 (2020). https://doi.org/10.1145/3371708, https://dl.acm.org/doi/10.1145/3371708
11. Grigorjew, A., Metzger, F., Hoßfeld, T., Specht, J., Götz, F.J., Chen, F., Schmitt, J.: Bounded latency with bridge-local stream reservation and strict priority queuing. In: 11th International Conference on Network of the Future, pp. 55–63 (2020). https://doi.org/10.1109/NoF50125.2020.9249224
12. Grigorjew, A., Seufert, M., Wehner, N., Hofmann, J., Hoßfeld, T.: Ml-assisted latency assignments in time-sensitive networking. In: IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 116–124 (2021)
13. Guck, J.W., Van Bemten, A., Kellerer, W.: Detserv: network models for real-time QOS provisioning in SDN-based industrial environments. IEEE Trans. Netw. Serv. Manage. **14**(4), 1003–1017 (2017). https://doi.org/10.1109/TNSM.2017.2755769. Dec
14. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95—International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995). https://doi.org/10.1109/ICNN.1995.488968
15. Le Boudec, J.Y., Thiran, P.: Network calculus: a theory of deterministic queuing systems for the Internet. Springer, Berlin, Heidelberg (2001). https://doi.org/10.1007/3-540-45318-0
16. Li, E., He, F., Zhao, L., Zhou, X.: A SDN-based traffic bandwidth allocation method for time sensitive networking in Avionics. In: 2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC), pp. 1–7 (2019). https://doi.org/10.1109/DASC43569.2019.9081700
17. Maile, L., Hielscher, K.S., German, R.: Network calculus results for TSN: an introduction. In: 2020 Information Communication Technologies Conference (ICTC), pp. 131–140. IEEE, Nanjing, China (2020). https://doi.org/10.1109/ICTC49638.2020.9123308
18. Maile, L., Hielscher, K.S.J., German, R.: Delay-guaranteeing admission control for time-sensitive networking using the credit-based shaper. IEEE Open J. Commun. Soc. **3**, 1834–1852 (2022). https://doi.org/10.1109/OJCOMS.2022.3212939
19. Maile, L., Voitlein, D., Grigorjew, A., Hielscher, K.S., German, R.: On the validity of credit-based shaper delay guarantees in decentralized reservation protocols. In: Proceedings of the 31st International Conference on Real-Time Networks and Systems. RTNS 2023, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3575757.3593644, forthcoming
20. Niemann, K.H.: PROFINET Design Guideline Version 1.53. Tech. Rep. 8.062, PROFIBUS Nutzerorganisation e.V., Karlsruhe, Germany (2022)
21. Soni, A., Scharbarg, J.L., Ermont, J.: Efficient configuration of a QoS-aware AFDX network with deficit round robin. In: 2020 IEEE 18th International Conference on Industrial Informatics (INDIN). vol. 1, pp. 251–258 (2020). https://doi.org/10.1109/INDIN45582.2020.9442115, iSSN: 2378-363X
22. Yen, J.Y.: Finding the K shortest loopless paths in a network. Manage. Sci. **17**(11), 712–716 (1971)