# Centralized vs. Decentralized: A Hybrid Performance Model of the TSN Resource Allocation Protocol

David Raunecker*, Stefan Geissler*, Alexej Grigorjew*, Philip Diederich†, Wolfgang Kellerer†, Tobias Hossfeld*
*Chair of Communication Networks, University of Würzburg, Germany
Email: {firstname.lastname}@uni-wuerzburg.de
†Chair of Communication Networks, Technical University of Munich, Germany
Email: {firstname.lastname}@tum.de

***Abstract*—Time-Sensitive Networking can provide a wide array of QoS guarantees that can be leveraged in a variety of use cases, ranging from industrial applications to autonomous driving. While the control plane can be configured following either a distributed or a centralized paradigm, it is difficult to make general assertions about the affinity of different topologies and use cases towards these configuration methods. In this work, we propose a hybrid simulation of the Resource Allocation Protocol to evaluate the reservation performance across a wide range of network configurations. Results show distinct behaviors between the configuration paradigms, making a case for decentralized control when scalability is critical. In addition, we make our implementation of the developed hybrid model publicly available.**

***Index Terms*—TSN, reservation, centralized, distributed, performance evaluation, simulation**

## I. Introduction

The emergence of Time-Sensitive Networking (TSN) has brought together the ubiquity of Ethernet with reliable real-time communication through standards ensuring deterministic latency, low jitter, and high reliability [1]. These features are vital for a wide range of applications, such as industrial automation, automotive networks, and audio-video bridging. TSN achieves deterministic latency, jitter, and packet loss bounds using mechanisms like priority queuing and resource reservation. Here, *resource reservation* acts on the control plane, managing traffic demands and configuring switches accordingly. Generally, the control plane can be either centralized or distributed [2, Sec. 46.1], where centralized controllers are the most prominent [3, Table 4]. Centralized approaches leverage a global network view and a dedicated controller to optimize resource allocation. Conversely, decentralized methods distribute the reservation process across multiple network nodes, promoting scalability and robustness. Each approach presents unique advantages and challenges, particularly in terms of complexity, scalability, and fault tolerance. However, the performance implications of these approaches in dynamic and heterogeneous network environments are not yet fully understood.

In centralized reservations, the speed of the controller significantly influences network performance. Faster controllers handle higher reservation rates and respond more quickly to network changes. However, the impact of controller speed on resource reservation efficiency, particularly in large-scale or high-traffic scenarios, requires detailed investigation. To this end, in this study, we evaluate the performance of the Resource Allocation Protocol (RAP) in centralized and decentralized modes across varying controller speeds and network topologies. TSN networks span diverse topologies—from simple linear chains to complex industrial networks—each posing unique challenges for resource reservation and traffic management. By analyzing multiple topological configurations, we explore the scalability of RAP in both centralized and decentralized settings and identify critical parameters affecting network efficiency. In order to mitigate the large increase in computation effort incurred by the simulation of the data-plane, we instead opt for a hybrid simulation model, simulating control-plane traffic while calculating delay in the control-plane due to data-plane traffic numerically rather than through simulation.

Understanding the details of RAP's performance in a wide range of network configurations, and across different network topologies, is essential for designing robust and efficient TSN networks. To this end, we make the following contributions in this work.

- We develop and implement a hybrid simulation and queuing theory model of the RAP protocol for both centralized and decentralized operation.
- We investigate the impact of centralized and decentralized operation on the performance of RAP across different topology sizes.
- We observe the impact of previously reserved streams on the computation and transmission time of subsequent reservations.

In addition, we make our implementation of the developed hybrid model publicly available.[1]

The remainder of this work is structured as follows: Section II provides background on TSN concepts and the Resource Allocation Protocol. Section III offers a brief overview

[1]https://github.com/lsinfo3/RAP-Simulator

of related work. Section V introduces the developed hybrid model, detailing its parameters and supplementary information on topology generation. Section V evaluates the simulation's functionality and compares centralized and decentralized operation. Lastly, Section VI concludes this work.

## II. Background

This section lays out relevant background for essential concepts regarding Time-Sensitive Networking, and the details of the Resource Allocation Protocol according to the standards defining them.

### A. Time-Sensitive Networking

Time-Sensitive Networking is a working group within the IEEE 802.1 Ethernet standardization group [4]. It encompasses standards focused on *synchronization*, *reliability*, *latency*, and *resource management*. Standards in the *latency* category enhance data plane queuing to ensure more predictable traffic behavior. Examples include priority transmission selection and Time-Aware Shaping for scheduled traffic [5, Sec. 8.6.8]. *Synchronization* is essential for scheduled traffic, *reliability* involves filtering and redundancy mechanisms, and *resource management* standards offer protocols and APIs for traffic demand signaling and switch configuration, serving as the core of TSN by providing stream information and configuration options for other categories.

In this work, Strict Priority queuing without shaping is utilized, meaning lower priority traffic can only be transmitted when higher priority queues are empty. Previous research [6] demonstrated deterministic latency guarantees without shaping, even in a decentralized control model, assuming end devices adhere to communicated traffic volumes and switches maintain configured latency thresholds. The worst-case latency formula was adapted to the token-bucket traffic model in RAP, detailed in pseudocode in the RAP standard [7, cf. *checkLatencyConstraintSP*]. While RAP supports various queuing methods beyond Strict Priority, this work does not consider other TSN standards, as latency models are not the focus here.

### B. Resource Allocation Protocol

Fundamentally, the Resource Allocation Protocol [7] (RAP) manages the distribution of traffic requirements and ensures pre-configured latency thresholds through admission control. RAP employs a publish-subscribe model, where *Talkers* advertise their data streams to the network, and *Listeners* subscribe to these streams. Figure 1 depicts this process with three end devices and one bridge.

In the first step, the Talker emits a *Talker Announce* (TA), including stream identification properties and a token bucket traffic model, which is an upper bound for the outgoing traffic volume. Each switch that receives this TA checks the available resources, including latency and bandwidth constraints. These checks must be performed for every individual egress port of the switch.

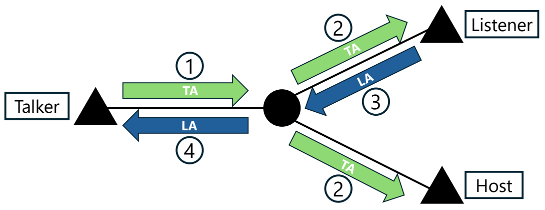


Figure 1: Illustration of the four main steps in RAP.

In step 2, the updated TAs are propagated to the next hops from all egress ports, with updated Failure Status and Accumulated Latency fields. The latter provide the current end-to-end latency guarantee and are required for the latency calculation during resource checks. If the latency calculations indicate that a stream cannot be accommodated due to latency thresholds, this is indicated in the Failure Status. Step two is repeated by each intermediate switch until every node in the network has received the TA.

In step 3, any host that receives the TA may become a Listener to that stream by subscribing to the TA. To this end, it transmits a *Listener Attach* (LA) through the port on which it received the corresponding TA.

In step 4, the LA is propagated back to its Talker. Here, the resource checks and latency computations must be repeated, because other stream reservations may have been accepted in the meantime due to race conditions. If a check fails, a flag is set on the outgoing LA, also updating the corresponding TA. Via a chain of updated TAs, the failure information can reach the Listener. However, if all resource checks pass, the Talker receives an LA with the *AttachReady* status and can begin its transmission.

In detail, TAs and LAs are not self-contained packets forwarded through the network. Instead, RAP uses the Link-Local Registration Protocol (LRP) [8], which provides small databases at each port where RAP can insert TA and LA records. LRP automatically synchronizes these records with neighboring ports, and the RAP stack is informed when new attributes arrive. This means, for example, multiple LA records from different ports may be consolidated into a single LA record at the egress, rather than triggering individual LAs at each hop. This behavior is replicated in our simulation, causing some unexpected effects during stream removal

In reality, a stream reservation can be canceled by removing the corresponding TA or LA attributes, which will also be propagated through the network. In our simulation, the LA status is changed to *Detached* instead in order to facilitate state tracking. The observable behavior from the outside remains unchanged.

## III. Related Work

Performance evaluation of TSN protocols touches multiple areas of literature. Most prominently, existing simulation frameworks [9], [10] focus on data plane simulations. They provide performance models for the delay experienced by data packets when using a particular shaper from the TSN tool set.

However, they do not capture the signaling and computation required to configure their scenarios.

Literature provides an excessive number of mathematical models to obtain worst case delay bounds [6], [11] and optimal schedules [12], [13] for TSN networks. They provide the foundation for implementations of signaling and configuration, such as this work. In contrast, the deployment of these configurations is rarely considered in literature [14], as many controller implementations remain undisclosed to the public. Some papers leverage the similarity of the Software-Defined Networking (SDN) architecture and the centralized TSN control model [15] to facilitate this deployment. Our previous work provided some performance insights into a decentralized publish-subscribe reservation protocol [16]. It showed that the controller performance decreased with increasing reservation count. However, to the best of our knowledge, there is no performance model, simulation, or public implementation of the TSN Resource Allocation Protocol (RAP).

Despite that, the centralized mode of operation bears noticeable resemblance to the SDN controller scenario. In this context, controller benchmarking tools [17]–[19] provide performance insights into SDN controllers without requiring a physical testbed. Further, considerations regarding controller placement [20], [21] and especially resilience [22], [23] could apply similarly in the TSN use case.

This work differs from the SDN control loop scenario, as different computations are involved in TSN resource reservations. With RAP, transmission times and computation delays for new stream reservations increase with each previously accepted stream. Additionally, a computation result can affect the processing time of future events; if a stream is rejected by one device, the others on the path skip unnecessary computations. RAP can also operate in centralized, decentralized, or hybrid modes per device. This complexity makes modeling performance challenging, prompting the development of this simulation.

## IV. Methodology

The following section details the methodology applied in this work. This includes a description of the developed simulation model, abstractions that discern the simulation from a real-world RAP protocol implementation, and surrounding mechanisms like topology and stream generation.

### A. Simulation Model Description

The simulation model abstracts arbitrary topologies using a graph representation where nodes are switches, or endpoint hosts, connected by edges as links. Each switch and host has a central CPU resource with an infinite queue. Ports have individual queues for different priorities (e.g., best effort, reserved streams) and operate in full duplex, allowing simultaneous transmission and reception. Figure 2 illustrates a single switch in the model. Control-plane traffic arrives at an ingress port, queues at the CPU for processing, and then moves to the appropriate exit ports TX queues. Simultaneously, data-plane traffic bypasses the CPU, directly entering the priority queue for its respective priority.

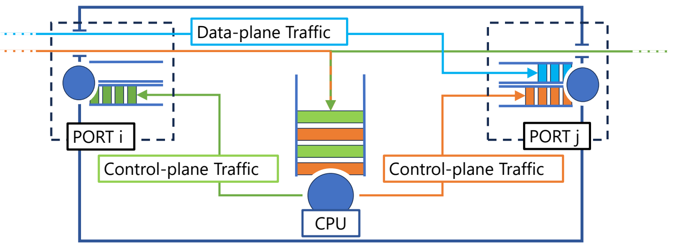


Figure 2: Switch model used within the RAP simulation.

CPU resources are defined by their processing speeds in operations per time unit. In the model, CPUs are responsible for resource checks and latency computations. The required operations (addition, subtraction, multiplication, division) for delay calculations depend on the number of previously accepted stream reservations, as each stream incurs a delay. After the CPU processes a control-plane packet, it forwards it to output ports where it is enqueued in TX queues. Transmission time depends on switch hardware and link speed. Thus, the end-to-end delay for control-plane packets includes CPU processing and queuing delays at switch output ports. Data-plane traffic bypasses the CPU, directly queuing at the TX port, sharing it with higher priority control-plane traffic. The transmission delay model accounts for data-plane traffic of reserved streams without explicit simulation. This hybrid approach is detailed below.

### B. Hybrid Delay Calculation

To keep complexity and runtime of the model evaluation in check, we developed a hybrid model, simulating control-plane packets individually, but abstracting data-plane traffic analytically, as we do not focus on the QoS of data-plane packets. The simulated computation time of RAP records is derived from the number of arithmetic operations required for latency computation. This number of operations is calculated according to Grigorjew et al. [6].

*1) CPU Computation Time:* Each TA and LA record must be briefly checked by the CPU before propagating to the next hop. If the reservation is already rejected or the Listener detaches, processing time is constant. However, if resource checks are needed, processing time depends on (i) the arithmetic operations for latency computation and (ii) the controller's clock speed. The number of operations is influenced by the latency model and the number of prior reservations, as each stream increases computation effort. Each network node has a single CPU unit; if busy, new records are queued in first-in, first-out order, as shown in Figure 2.

*2) Record Transmission Time:* Transmissions are commonly subject to four types of latency. (i) Propagation delay is negligible in a LAN topology due to short distances. (ii) Processing delays for the records are handled separately. (iii) Transmission delay is given by packet size divided by link speed. Here, all records are simply assumed to be 120 B.

(iv) Queuing delay depends on other records currently being transmitted, and on higher priority data-plane packets from admitted streams. To avoid simulating the data-plane, we combine simulation and analytical methods in a hybrid model.

To this end, an M/M/1 queuing theory model is employed to obtain the waiting time distribution of data-plane packets in the steady state. Inter-arrival times of data-plane packets are assumed to be i.i.d. in accordance to the Palm-Khintchine theorem [24]. Service times primarily depend on the packet size of data-plane packets, also assumed to be exponentially distributed, without loss of generality. Note that these assumptions concern the data plane traffic. If other traffic models do not meet these assumptions, more sophisticated models for waiting times in general systems can be implemented instead.

For such a system, we can calculate, for each specific port of any switch, the total arrival rate $\gamma_l$ as the sum of rates from the individual data-plane streams $s \in \mathcal{S}_l$ on port $l$: $\gamma_l = \sum_{s\in\mathcal{S}_l} \mathit{rate}_s$. Aiming for a packet-based analysis, we are interested in the superposition of all processes describing the distribution of packet sizes for the data-plane streams running on port $l$. As these processes are assumed to follow an exponential distribution, this also follows an exponential distribution, with the scale parameter $\phi_l$ the weighted sum of each stream's scale parameter $\mathit{burst}_s$: $\phi_l = \left(\sum_{s\in\mathcal{S}_l} \mathit{burst}_s \cdot \mathit{rate}_s\right)/\gamma_l$. With this, we can compute the packet arrival rate as $\alpha_l = \gamma_l/\phi_l$, and the average packet inter-arrival time as its inverse $\iota_l = \alpha_l^{-1}$.

Modeling the data-plane traffic as an M/M/1 waiting system mainly offers the advantage that the waiting time distribution in the steady state is known. Based on this, we can draw the current queue size of the data-plane traffic on arrival of a control plane packet from this known queue size distribution, which means it is not necessary to simulate the data-plane traffic in its entirety. However, the M/M/1 model only includes data-plane traffic, without the TA and LA records from the simulation. As the waiting time distribution is only valid in the steady state, two different cases must be considered.

First, the last TA/LA record has been fully transmitted sufficiently long ago that data-plane traffic can be assumed to be in a steady state. As observed above, the total packet arrival rate $\alpha_l$ can be calculated due to assumptions about the arrival process. Additionally, the assumption that packet sizes are exponentially distributed allows for the calculation of the packet service rate $\beta_l$ at a link with bandwidth $\mathit{bw}_l$: $\beta_l = \mathit{bw}_l/\phi_l$. Now, the utilization $a_l$ of the queue modeling the port $l$ is calculated through $a_l = \alpha_l/\beta_l$. With $a_l$ known, the steady-state queue size distribution can be calculated.

Considering the different priority classes, control-plane packets are assigned lower priority to avoid interfering with the strict time constraints of the data-plane. Thus, even if the data-plane queue size is known when a control-plane packet arrives, any new data-plane packets will overtake it. Given the Markovian nature of data-plane traffic and service times, we can subtract the packet arrival rate from the service rate to find the effective service rate for control-plane packets. This allows the computation of the total waiting time for a control-plane packet due to data-plane packets without needing to explicitly simulate data-plane packets. However, this assumes the data-plane queue is in steady state upon the control-plane packet's arrival, which is not always true.

In the second case, a steady state cannot be assumed. As the queue size at the arrival of a control-plane packet $c$ does not follow a known distribution, we simulate it instead. Since the queue containing data-plane packets implies a short time since the last control-plane packet was serviced, computation costs are relatively low. The simulation involves tracking the arrival and processing of data-plane packets since the last control-plane packet before $c$, when the data-plane queue was empty, to determine the queue size at $c$'s arrival. Once this is calculated, the waiting time of $c$ can be computed by determining the transmission time for all data-plane packets in the queue, assuming a service rate reduced by the arrival rate of new data-plane packets, as described in the previous case.

To determine if enough time passed for the data-plane queue to return to a steady state, multiple models or heuristics could be employed. In this work, we decide on a simple threshold of ten times the data-plane traffic packet inter-arrival time $\iota_l$. In that case, the steady-state assumptions is valid and the numerical error is below a certain threshold, which was verified in our results. Due to space constraints, we have omitted these results here.

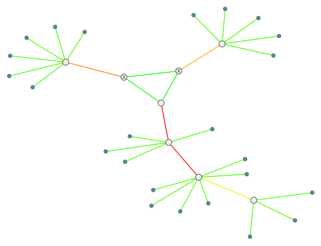
## C. Centralized vs Decentralized Operation

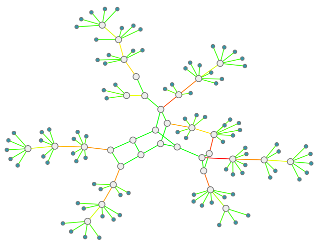
RAP can be implemented in two ways: Centralized or decentralized. In the centralized case, at least one controller node must be present in the network, performing necessary calculations to guarantee adherence to pre-defined latency bounds for each stream. For this paper, we define two distinct centralized configuration methods, and compare them to the decentralized one.

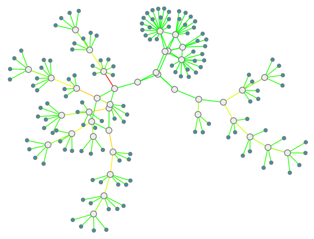
In the first centralized method, a controller connects to all switches in the network with designated links, creating a management network fully isolated from data-plane traffic. This allows for communication between switches and the centralized controller without control-plane and data-plane traffic interfering with each other, at the cost of having to provide and maintain the dedicated management network. Going forward, we call this method *centralized extra*.

For the second mode of operation, the controller is placed within the network, specifically at the site of one of the switches, and is only connected to its colocated switch. Thus, control-plane messages are transmitted through the network that also carries data-plane traffic to the switch for processing. Thereby, control-plane and data-plane traffic can interfere with each other, reducing the overall performance of the reservation process, but eliminating the need for a dedicated management network. In the following, we call this method *centralized intra*.

Finally, during decentralized operation, no dedicated control instance exists. Instead, every switch performs its own control-plane processing without the need for communication to a controller. Decentralized operation is expected to scale better


(a) Small topology.


(b) Medium topology.


(c) Big topology.

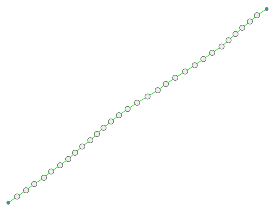

(d) Linear Topology.

Figure 3: Example topologies for different classes.

with the number of nodes in the network. However, we know from literature that the control-plane capabilities of switches exhibit strong performance limitations [25]. Going forward, we call this method *decentralized*.

## D. Topology Generation

Given the significant impact of topology on stream reservation performance, we evaluate RAP across a range of topologies. To properly assess the effects of different topologies, we generate topologies mimicking industrial networks. These networks typically consist of a central ring with branching linear chains connected to it. Endpoints, i.e. sensors or actuators in industrial settings, are connected to these branches; we refer to these nodes as hosts. We categorize the topologies into three classes (small, medium, big), corresponding to the number of nodes and links, and slightly varying structure. Figure 3 shows examples of each topology class and a simplified topology with a single line of 30 switches and hosts at the endpoints.

## E. Simulation Parameters

In addition to the topology, the streams for which the RAP protocol allocates resources significantly influence system behavior. Each stream involves one host advertising its data stream to the network and one predefined listener responding positively. While the protocol and simulation support multiple listeners, we limit it to one per stream to manage the parameter range. Each stream is characterized by a rate, the amount of data (in bits) allowed to transmit over a time interval, and a burst size, the maximum bits transmitted at once. For our purposes, the burst size equals the maximum packet size for the stream.

In order to perform latency calculations for each switch, absolute latency thresholds for each switch in the network need to be configured in advance. In this work, we have chosen to apply uniform latency guarantees for each switch. Mainly, this is meant to simplify the parameter space, as there is an infinite amount of different configurations which may skew the simulation results.

## F. Switch Performance

The speed at which a switch's CPU can perform operations and determine if latency bounds for a certain link and priority can be met is crucial for analyzing the RAP's performance. Based on [16], we estimate the number of operations a switch can perform. Their analysis indicates that for 300 reserved streams, 55 streams can be processed per second, and for 800 reserved streams, 25 reservations can be processed per second. This suggests that each stream adds approximately 60 µs of processing time at 300 streams and 50 µs at 800 streams. Given that processing one stream requires about 50 operations, we set the service rate of switch CPU resources to approximately $1e6$ operations per second.

In order to estimate the number of operations that a controller node in the centralized configuration can perform, we are currently not aware of any measurements or estimates, and these could differ significantly between different use-cases. For now, we generally consider the clock speed of a controller to be 100 times faster than a regular switch. However, we perform an evaluation of different clock speeds for the centralized controller, in order to compare the performance of centralized and decentralized configuration methods with regard to different performance capacities.

Note that when applying the simulation for a concrete use-case, parameters of switches and controllers need to be fine-tuned to gain more fine-grained insight into the concrete application. For now, the application to specific use-cases is beyond the scope of this work, but may be refined with concrete technical details of realistic TSN networks in the future.

## V. Evaluation

The following section provides an evaluation of both the functionality of the simulation and a comparison of different performance measures between the centralized and decentralized configuration methods.

## A. Linear Scenario with Homogeneous Streams

To identify the core reasons of observed delays in the system, we begin with a simple scenario. A linear topology of 30 switches with hosts at the endpoints, as shown in Figure 3d, where all streams flow from one endpoint to the other. All streams are assumed to have identical technical specifications (e.g., maximum bit rate and burst size). Evaluating the simulation in this straightforward setup allows us to isolate specific patterns and phenomena that may be overlooked otherwise.

## B. Simulation Functionality

Before studying RAP's reservation performance, it is important to qualitatively evaluate the simulation behavior for consistency with the model. Figure 4 shows the total delay for different reserved streams (x-axis), with delay components indicated by color (y-axis), based on the linear topology in

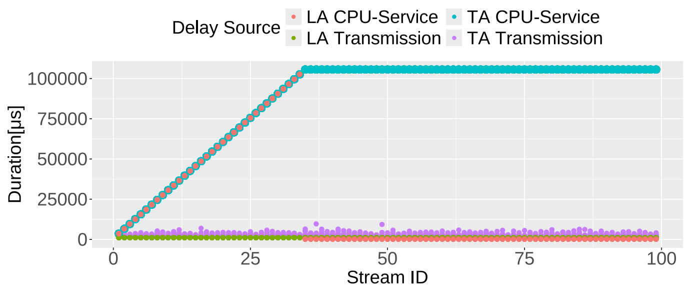


Figure 4: Delay composition for TAs and LAs in subsequent streams.

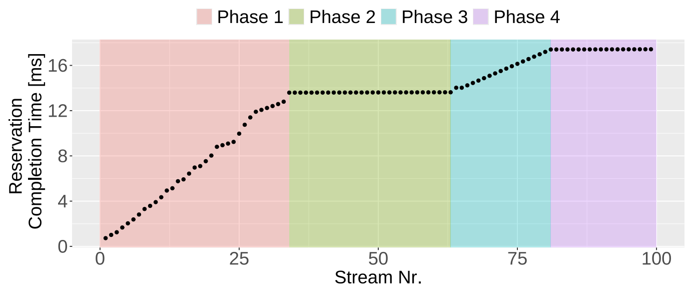


Figure 5: Total end-to-end reservation time of subsequent streams in linear topology.

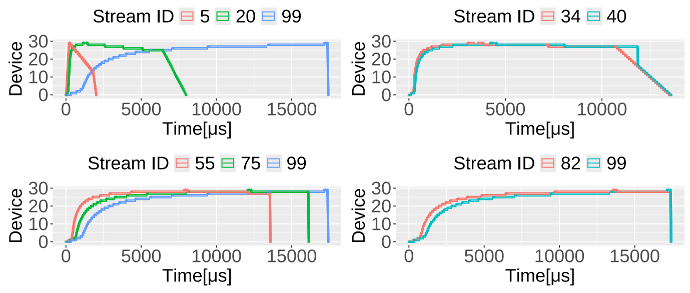


Figure 6: Propagation of TA and corresponding LA through a linear topology.

Figure 3. The time between stream reservations is kept high to avoid control-plane packet interactions, isolating delay sources for analysis. Only the decentralized configuration is considered here, as delay sources are consistent across all three methods.

Firstly, the data shows no delays due to queuing at ports or CPU resources, which aligns with our aim to avoid stream interactions. Hence, this data is excluded from the plot as it would be constantly zero. In Figure 4, CPU processing times for TAs and LAs (blue and red dots respectively) show a linear increase for the first 34 streams. This increase matches the expected scaling of delay calculations with the number of previously reserved streams. From the 35th stream onward, CPU-Check for TAs shows a constant total delay, while for LAs, the delay drops to almost zero. The 35th TA hits a delay threshold, preventing its placement and halting further delay calculations, so no additional delay is incurred. Consequently, the LA response for these streams incurs almost no delay, staying close to zero.

Regarding transmission times for TA and LA, both are similarly small, indicating that, even with the data-plane traffic being considered, the transmission time is negligible compared to the time required for latency calculations. These results are, however, dependent on link bandwidths, stream properties, and CPU clock speeds, and are meant to prove the functionality of the simulation, not to illustrate the real-world relationship between latency sources.

With basic functionality qualitatively assured, we focus on delays in the same linear topology to provide explanations for how these delays emerge. However, we now investigate a scenario in which the Talker enqueues all TAs at the same time, leading to interactions within control-plane traffic.

*1) Detailed Delay Origin Analysis:* To better understand delays in the proposed simulation model, this section continues with the previously introduced linear topology with the decentralized configuration method. However, unlike the earlier, all reservations now start simultaneously, causing control-plane packet interactions and queuing delays. While these delay patterns could theoretically be deduced using queuing theory, the observations are instructive for understanding the delays, as the principles explaining them generalize to more complex systems.

We placed 100 streams in the given topology and tracked the timing of the last event in each reservation process. This event usually occurs when the LA reaches the Talker (the device advertising the stream) or when it informs the Listener of a failed attachment due to insufficient network resources. Figure 5 shows these results, with stream placement order on the x-axis and the last event time on the y-axis, revealing four distinct patterns. To explain these, we examine delays at certain switches in detail. Figure 6 shows the order switches are traversed by TAs on the y-axis, with the x-axis representing the sojourn time of each TA/LA. For each stream (indicated by different colors), a TA travels from switch 0 to switch 30, then an LA returns. This visualization details delays at each switch, explaining the patterns in Figure 5.

1) For the first 34 Streams, reservations can be completed normally. It can be seen that there is an approximately linear upward trend, meaning that the time between completed reservations is fairly similar. This behavior is perturbed by other TAs and LAs in the system, but is relatively stable. An example of this type are streams 5 and 20 in Figure 6.

2) From Stream 35 to 63, there is no increase in the reservation completion time for consecutive streams. For Stream 35 and consecutive streams, further reservations cannot be made, as it would violate the delay threshold. Thus, switches need not make any computations when checking LAs, and simply pass them through. Due to FIFO, all of these LAs then wait behind the 34th LA, for which checks are still performed, and thus match its trajectory. Stream 40 in Figure 6 illustrates this.

3) After Stream 64, the time to full reservation completion for each stream increases again. This matches the behavior of the first 34 streams, but with fewer streams causing congestion, the increase is less steep. Between Streams 55 and 75 in

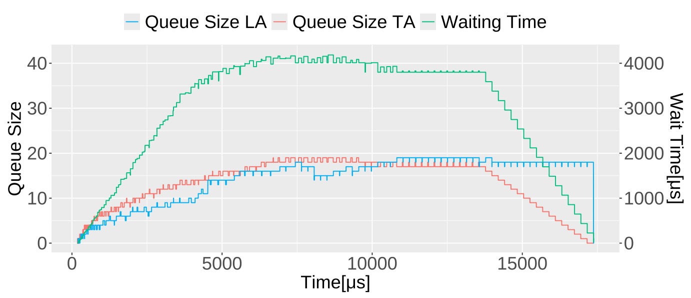


Figure 7: Queue size and total delay for packets at switch 29

Figure 6 the full reservation completion time increases.

4) Lastly, Streams 81 and after experience a delay as they all arrive at switch 29 after the last TA of Stream 99 has already enqueued, meaning they wait for this TA, and thus all TAs, to be processed. After the TA of Stream 99 has passed this bottleneck, all LAs can quickly move to the Talker unencumbered, and all the control-plane messages of all streams arrive at their origin near simultaneously. This is observed on the bottom right of Figure 6 for Streams 82 and 99.

Summarizing insights from Figure 6, it becomes evident that latency patterns arise from interactions between streams, especially near the Listener in the network, i.e., switch 30. Figure 7 illustrates both the number of TAs and LAs in the single CPU-resource queue (in red and blue) and the packet waiting time (in green) for switch 29. Although queue size and waiting time use different y-axis scales, there is a strong correlation of about $95.59\%$, indicating that waiting times at this switch are primarily due to CPU-resource queuing delays.

With the fundamental latency sources and simulation behavior qualitatively assessed, the remainder of the evaluation concentrates on the comparison between distributed and centralized configuration methods for the RAP simulation.

## *C. Bulk Reservation Scenario*

To generalize RAP's performance using centralized or decentralized configurations, realistic network topologies are needed. Lacking an extensive evaluation library, we created randomized topologies typical of industrial use-cases. We generated 150 topologies with random technical specifications for $1,000$ streams, with start and endpoints chosen uniformly at random. Stream parameters are omitted as they are widely irrelevant to the analysis. The 150 topologies include 50 each of small, medium, and large sizes, differing mainly in node count and slightly in structure to fit different network sizes.

The bulk reservation scenario involves all reservations starting simultaneously at the start of the simulation, with Talkers enqueuing all TAs at once. This simulates a total reconfiguration of an industrial application, possibly after a power outage or a fundamental change in production mode. We focus on the time required to complete all reservations.

*1) Performance Impact of Centralized Controller:* To compare the three configuration methods—centralized extra, centralized intra, and decentralized—we focus on the time required to fully configure the network by exchanging all control-plane messages to reserve all streams. This section characterizes the performance of the protocols at different clock speeds for the centralized controller in both centralized scenarios and compares it to the decentralized case, particularly regarding scalability for larger network sizes.

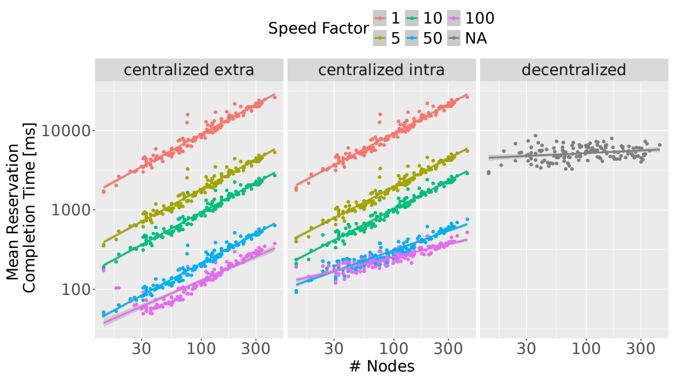


Figure 8: Mean time required for all reservations to complete in the bulk reservation scenario, on a double logarithmic scale.

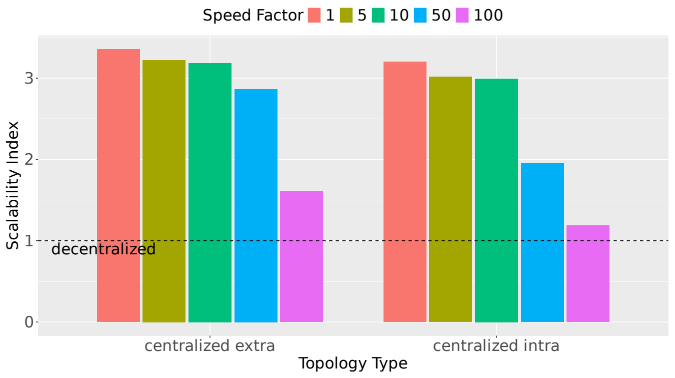


Figure 9: Scalability Index for normalized MRCP as target measure by Topology Type and Speed Factor, using the decentralized configuration method as Reference System.

Clock speed is varied relative to the base clock speed of each switch in the network, set at $1,000,000$ operations per second. The central controller's clock speed can be $1, 5, 10, 50$, or $100$ times faster than this base speed. This Speed Factor parameter does not impact the decentralized method. To ensure statistical reliability, each parameter combination is repeated 10 times.

The simulation results in Figure 8 display the time to configuration completion (y-axis) across different network sizes (x-axis) for each configuration method. Colors indicate the Speed Factor. Due to variations in completion times, both the y-axis and x-axis are logarithmically scaled. Each parameter combination includes both raw data and a linear regression model.

Comparing the decentralized approach to the two centralized variations, even with a fivefold Speed Factor, the centralized methods exhibit superior performance in smaller topologies. However, the centralized methods perform worse as the number of nodes increases, despite consistently placing 1,000 streams in each simulation run across varying topologies. Larger topologies distribute streams more widely, effectively increasing the total processing capacity in the decentralized scenario, which centralized variants do not leverage similarly. This indicates that decentralized approaches might scale better to larger network sizes.

To investigate the scalability of different Topology Types and Speed Factors with respect to Mean Reservation Completion Time (MRCT) as the target measure, we compute the Scalability Index with the number of nodes as the discrete input parameter, following Hoßfeld et al. [26]. To focus on scalability rather than absolute performance, we normalize values within each Topology Type and Speed Factor group by the lowest MRCT. This is shown in Figure 9, using the decentralized configuration method as the reference system, denoted by the dotted horizontal line. The figure indicates that, overall, centralized variants scale better for higher Speed Factors, but none match the SI of the centralized configuration. Future work could examine decentralized behavior with larger networks and varying stream numbers to see if these results hold in broader scenarios.

Comparing the centralized configuration variants for constant Speed Factors, it's evident their performance is similar. This is because, under the current configuration parameters, processing delays dominate transmission delays. Whether the controller uses an external network or shares the same network as data-plane traffic, the processing delay remains unchanged, resulting in comparable performance for both methods.

However, at higher clock speeds, transmission delays become more significant. This is notably observed in Figure 8, especially with a Speed Factor of 100. In such cases, the centralized extra configuration shows advantages over the intra method, particularly in smaller topologies where these effects are amplified. This understanding is crucial for network design, where dedicating separate links for control-plane traffic can increase costs but potentially improve performance depending on the balance between transmission and processing delay, as well as network size.

While these insights into the behavior for a bulk reservation behavior can be helpful, it only covers part of the range of TSN use-cases. One advantage of TSN lies in its flexibility, leveraging the ability to reconfigure a network on the fly.

### *D. Dynamic Reservation Scenario*

The dynamic scenario aims to evaluate RAP's capability to handle continuous stream reservation and de-reservation over time. To achieve this, 1000 streams are gradually introduced into the topologies using various arrival rates with negative exponentially distributed inter-arrival times. Based on this, we set the mean sojourn time of streams — the time a stream remains in the system — based on Little's law, so that, on average, 100 streams are reserved concurrently.

*1) System Load:* To evaluate system performance in this scenario, we analyze switch CPU utilization. Specifically, for centralized methods, we examine the controller's CPU utilization, while in the decentralized approach, we consider the maximum CPU utilization across all switches, as an indicator of the load of the total system. The controlling switches are configured with a Speed Factor of 100 over regular switches. For the evaluation, we discard transient phases from simulation runs and calculate the CPU utilization as the fraction of active CPU time during the steady state relative to total simulation time in the steady state. Figure 10 displays the maximum CPU utilization for each configuration method. Box plots represent the range of values across various topology sizes and 10 repetitions, plotted on the y-axis with a logarithmic scale. Across increasing mean inter-arrival times on the x-axis, the performance between the two centralized configuration methods remains indistinguishable.

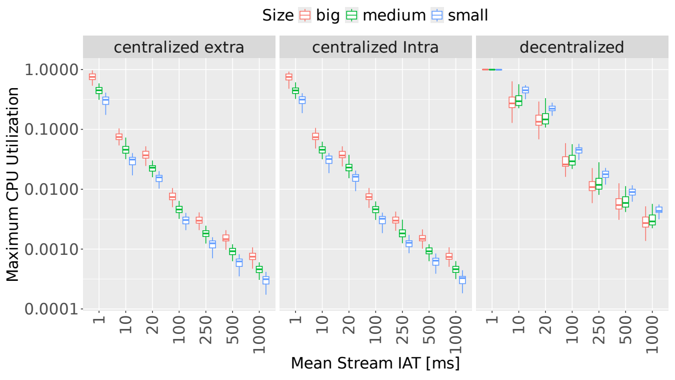


Figure 10: CPU utilization of controlling node in the dynamic reservation scenario with speed increase of 100 for the centralized controller.

In the decentralized case, with a mean inter-arrival time of 1ms, the system shows switches at 100% utilization, with at least one critical switch's CPU remaining active throughout the steady state phase. Although this switch's CPU utilization exceeds that of controlling nodes in centralized scenarios, it does not necessarily imply inferiority of the decentralized method in this metric, given the substantially more powerful CPUs used in the centralized methods.

Interestingly, CPU utilization decreases in decentralized, larger topologies, whereas it increases for centralized methods. This likely results from a more evenly distributed load across more switches in larger networks, whereas centralized setups face increased computational demands with larger topologies. While the centralized configuration methods outperform the centralized one even with minor speed increases on the present topologies, its scalability suggests that the decentralized method may be more efficient in distributed settings, potentially reducing bottlenecks in sufficiently large deployments. While beyond the scope of this work, exploring this characteristic further, e.g., comparing performance by suitable graph metrics, remains future work.

## VI. Conclusion

Through the integration of Ethernet with novel mechanisms enabling real-time communication, TSN is paving the way for a new generation of applications, operating under hard real-time constraints. By rigorously allocating resources to specific traffic streams, TSN performs strict resource reservation to ensure consistent QoS across a wide range of networks.

To further our understanding of resource reservation in TSN, we have developed and evaluated a hybrid performance model for the Resource Allocation Protocol, which takes into account both control-plane and data-plane traffic. By combining queueing theory and discrete event simulation, we implement a comprehensive performance model and conduct a case study comparing the reservation performance in centralized and decentralized network configurations.

Our evaluation indicates that comparing two centralized approaches — one using a dedicated management network and the other sharing the network with data-plane traffic — reveals minimal differences in performance characteristics. This is primarily due to low transmission delay impact, influenced by link speeds and data-plane packet rates. However, with high-performance controllers, reduced processing delays amplify the significance of transmission delays. In such scenarios, a dedicated management network outperforms shared networks by processing 1,000 bulk reservations more efficiently. In addition, the data has shown that slight performance advantages of a centralized controller over switches resulted in centralized variants achieving better reservation completion times. When considering de-reservations, our analyses showed that decentralized control performs better for large topologies, while the opposite is true for centralized control.

While the conclusions drawn in this work are useful, we are still lacking concrete heuristics on graph properties that affect the affinity of the corresponding network topology to the investigated configuration methods, which we aim to investigate in a future work. Additionally, we are working on a TSN testbed, including a RAP implementation. Lastly, we plan on modeling data- and control-plane traffic thorugh a priority queuing model, in order to gain higher-level insight into traffic patterns that can emerge in TSN networks.

Finally, to the best of our knowledge, we make the first performance model of the Resource Allocation Protocol publicly available to provide the research community with the tools to perform further investigations in this direction.

## Acknowledgement

This work was funded by the Deutsche Forschungsgesellschaft DFG (German Reasearch Agency) under Grant No 316878574, project SDN-App.

## References

[1] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, *et al.*, "Ultra-Low Latency (ULL) Networks: The IEEE TSN and IETF DetNet Standards and Related 5G ULL Research," *IEEE Communications Surveys & Tutorials*, 2019.

[2] "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.

[3] Z. Satka, M. Ashjaei, H. Fotouhi, M. Daneshtalab, M. Sjödin, and S. Mubeen, "A comprehensive systematic review of integration of time sensitive networking and 5G communication," *Journal of Systems Architecture*, 2023.

[4] *Official website of the IEEE 802.1 Time-Sensitive Networking task group*, https://1.ieee802.org/tsn/.

[5] "IEEE Standard for Local and Metropolitan Area Network – Bridges and Bridged Networks," *IEEE Std 802.1Q (Revision of IEEE Std 802.1Q-2014)*, 2018.

[6] A. Grigorjew, F. Metzger, T. Hoßfeld, *et al.*, "Bounded Latency with Bridge-Local Stream Reservation and Strict Priority Queuing," in *11th International Conference on Network of the Future (NoF)*, Nov. 2020.

[7] "IEEE Standard for Local and Metropolitan Area Networks – Bridges and Bridged Networks – Amendment: Resource Allocation Protocol (RAP)," *IEEE Std P802.1Qdd D0.9*, 2024.

[8] "IEEE Standard for Local and Metropolitan Area Networks – Link-local Registration Protocol," *IEEE Std 802.1CS-2020*, 2020.

[9] A. Grigorjew, F. Metzger, T. Hoßfeld, and J. Specht, "A Simulation of Asynchronous Traffic Shapers in Switched Ethernet Networks," in *2019 International Conference on Networked Systems (NetSys)*, 2019.

[10] J. Falk, D. Hellmanns, B. Carabelli, *et al.*, "NeSTiNg: Simulating IEEE Time-sensitive Networking (TSN) in OMNeT++," in *2019 International Conference on Networked Systems (NetSys)*, 2019.

[11] L. Maile, K.-S. Hielscher, and R. German, "Network Calculus Results for TSN: An Introduction," in *2020 Information Communication Technologies Conference (ICTC)*, 2020.

[12] S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner, "Scheduling Real-Time Communication in IEEE 802.1Qbv Time Sensitive Networks," in *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, ser. RTNS '16, Brest, France: Association for Computing Machinery, 2016.

[13] N. G. Nayak, F. Dürr, and K. Rothermel, "Incremental Flow Scheduling and Routing in Time-Sensitive Software-Defined Networks," *IEEE Transactions on Industrial Informatics*, 2018.

[14] B. Houtan, A. Bergström, M. Ashjaei, M. Daneshtalab, M. Sjödin, and S. Mubeen, "An Automated Configuration Framework for TSN Networks," in *2021 22nd IEEE International Conference on Industrial Technology (ICIT)*, 2021.

[15] S. B. H. Said, Q. H. Truong, and M. Boc, "SDN-based configuration solution for IEEE 802.1 time sensitive networking (TSN)," *SIGBED Rev.*, Feb. 2019.

[16] A. Grigorjew, C. Baier, F. Metzger, and T. Hoßfeld, "Distributed Implementation of Deterministic Networking in Existing Non-TSN Ethernet Switches," in *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2021.

[17] M. Jarschel, C. Metter, T. Zinner, S. Gebert, and P. Tran-Gia, "OFCProbe: A platform-independent tool for OpenFlow controller analysis," in *2014 IEEE Fifth International Conference on Communications and Electronics (ICCE)*, 2014.

[18] L. Zhu, M. M. Karim, K. Sharif, *et al.*, "SDN Controllers: A Comprehensive Analysis and Performance Evaluation Study," *ACM Comput. Surv.*, Dec. 2020.

[19] S. Herrnleben, P. Rygielski, J. Grohmann, S. Eismann, T. Hoßfeld, and S. Kounev, "Model-based performance predictions for SDN-based networks: A case study," in *Measurement, Modelling and Evaluation of Computing Systems: 20th International GI/ITG Conference, MMB 2020, Saarbrücken, Germany, March 16–18, 2020, Proceedings 20*, Springer, 2020.

[20] S. Lange, S. Gebert, T. Zinner, *et al.*, "Heuristic approaches to the controller placement problem in large scale SDN networks," *IEEE Transactions on Network and Service Management*, 2015.

[21] T. Das, V. Sridharan, and M. Gurusamy, "A Survey on Controller Placement in SDN," *IEEE Communications Surveys & Tutorials*, 2020.

[22] P. Vizarreta, C. M. Machuca, and W. Kellerer, "Controller placement strategies for a resilient SDN control plane," in *2016 8th International Workshop on Resilient Networks Design and Modeling (RNDM)*, 2016.

[23] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia, "Pareto-optimal resilient controller placement in SDN-based core networks," in *Proceedings of the 2013 25th International Teletraffic Congress (ITC)*, 2013.

[24] C. Palm, "Intensitätsschwankungen im Fernsprechverker," *Ericsson Technics*, 1943.

[25] A. Nguyen-Ngoc, S. Lange, S. Geißler, T. Zinner, and P. Tran-Gia, "Estimating the Flow Rule Installation Time of SDN Switches when Facing Control Plane Delay," in *International Conference on Measurement, Modelling and Evaluation of Computing Systems*, 2018.

[26] T. Hossfeld, P. E. Heegaard, and W. Kellerer, "Comparing the scalability of communication networks and systems," *IEEE Access*, vol. 11, pp. 101 474–101 497, 2023. DOI: 10.1109/ACCESS.2023.3314201.