HERMES: Heuristic Multi-queue Scheduler for TSN Time-Triggered Traffic with Zero Reception Jitter Capabilities

Daniel Bujosa
daniel.bujosa.mateu@mdu.se
Mälardalen University
Västerås, Sweden

Mohammad Ashjaei
mohammad.ashjaei@mdu.se
Mälardalen University
Västerås, Sweden

Alessandro V. Papadopoulos
alessandro.papadopoulos@mdu.se
Mälardalen University
Västerås, Sweden

Thomas Nolte
thomas.nolte@mdu.se
Mälardalen University
Västerås, Sweden

Julián Proenza
julian.proenza@uib.es
University of the Balearic Islands
Palma, Spain

ABSTRACT

The Time-Sensitive Networking (TSN) standards provide a toolbox of features to be utilized in various application domains.The core TSN features include deterministic zero-jitter and low-latency data transmission and transmitting traffic with various levels of time-criticality on the same network. To achieve a deterministic transmission, the TSN standards define a time-aware shaper that coordinates transmission of Time-Triggered (TT) traffic. In this paper, we tackle the challenge of scheduling the TT traffic and we propose a heuristic algorithm, called HERMES. Unlike the existing scheduling solutions, HERMES results in a significantly faster algorithm run-time and a high number of schedulable networks. HERMES can be configured in two modes of zero or relaxed reception jitter while using multiple TT queues to improve the schedulability. We compare HERMES with a constraint programming (CP)-based solution and we show that HERMES performs better than the CP-based solution if multiple TT queues are used, both with respect to algorithm run-time and schedulability of the networks.

CCS CONCEPTS

• **Networks** → **Packet scheduling**; • **Computer systems organization** → **Real-time systems**;

KEYWORDS

TSN, time-triggered, scheduling, zero jitter, heuristics.

ACM Reference Format:
Daniel Bujosa, Mohammad Ashjaei, Alessandro V. Papadopoulos, Thomas Nolte, and Julián Proenza. 2022. HERMES: Heuristic Multi-queue Scheduler for TSN Time-Triggered Traffic with Zero Reception Jitter Capabilities. In *Proceedings of the 30th International Conference on Real-Time Networks and Systems (RTNS '22), June 7–8, 2022, Paris, France.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3534879.3534906

RTNS '22, June 7–8, 2022, Paris, France

ACM ISBN 978-1-4503-9650-9/22/06.
https://doi.org/10.1145/3534879.3534906

1 INTRODUCTION

Data communication in industrial systems has dealt with many challenges during recent years, such as scalability in data transmission, high volume of data exchange, the coexistence of diverse applications with different time-criticality requirements, and guaranteeing deterministic transmission for hard real-time traffic. These challenges are mainly due to recent demands for increasing functionalities in industrial systems that impose further pressure on the data communication design of such systems. For instance, in several application domains, e.g., autonomous vehicles and smart automation, many sophisticated smart sensors and cameras are utilized to perform newly added functionalities that require a high amount of communication bandwidth and at the same time meet their timing requirements. Besides the timing requirements, the rise of adaptive industrial systems imposes another criterion for designing data communication systems in which the network should be reconfigured due to changes in the environment. Therefore, in such systems, the configuration of the network is not seen as a one-time configuration in the initialization phase, but as a dynamic reconfiguration during the run-time (and operational) phase.

IEEE Audio-Video Bridging (AVB) Task Group (TG) was established in 2005 to provide Ethernet with soft real-time capabilities oriented to audio/video streaming. The three main projects developed by this TG are: (i) the IEEE Std 802.1AS [3] for clock synchronization, (ii) the IEEE Std 802.1Qav, which standardized the Credit-Based Shaper (CBS) [1]; and finally (iii) the IEEE Std 802.1Qat, which standardized the Stream Reservation Protocol (SRP) [2]. The latter standard is particularly interesting in the context of dynamic networks as it allows adding and removing streams at run-time. As the features that were developed by the AVB TG became relevant to other application areas, such as automotive [23], automation [27], and energy distribution [22], new requirements emerged. Therefore, in 2012, the TG broadened its objectives to meet the demands and was renamed to Time-Sensitive Networking (TSN) TG. Specifically, TSN TG's work was developed as a set of standards to provide transmission of hard and soft real-time traffic on the same network, deterministic zero-jitter and low-latency transmission, precise clock synchronization, fault tolerance mechanisms, and advanced network management allowing dynamic reconfiguration.

Motivation: One of the main features developed within TSN TG is the zero-jitter traffic transmission, known as the Time-Aware Shaper (TAS), which is particularly utilized in applications that

require low-latency and low-jitter data transmission, e.g., in embedded control systems. The TAS allows transmission of Time-Triggered (TT) traffic while preventing any interference from other traffic via a gate mechanism on the ports of the switches. Therefore, TAS requires the synthesis of the Gate Control Lists (GCL) that are specifying at which point in time each frame should be transmitted. A GCL is defined for each switch port which contains 8 queues, in such a way that the GCL identifies the moments in which the gate of each queue will be open. The scheduling of TT traffic, and its synthesis in GCLs, is known to be an NP-complete problem [19]. Several solutions are proposed in the literature to schedule TT traffic in TSN networks that are mainly based on Integer Linear Programming (ILP) and Constrained Programming (CP) [7]. These solutions are known to have high time complexity, i.e., they require a long time to schedule large networks, thus they are not generally scalable. In addition, these solutions are not suitable for systems that require dynamic reconfigurations as the new configuration should be created relatively fast. Few heuristic schedulers are also proposed, e.g., [17], whose performance is not properly compared with the ILP and CP solutions.

Paper contributions: In this paper, we propose a heuristic scheduler for TT traffic in TSN networks, called Heuristic Multi-queue Scheduler (HERMES), that takes advantage of multiple queues for TT traffic to provide high schedulability with very low scheduling times. Frames in HERMES can be configured to be scheduled in two modes of zero or relaxed reception jitter, which provides better control for users. Through a set of experiments, we show that HERMES can perform better than CP-based solutions, i.e., it results in more schedulable networks, by allowing it to use multiple queues, and at the same time, it provides the results within 17 to 800 times faster. In our experiments with two sizes of networks, we obtained schedules in less than $1ms$, which shows that HERMES is suitable for dynamic reconfiguration of networks.

Paper outline: The paper is organized as follows. Section 2 presents the related work. Section 3 presents the background. Section 4 presents the proposed algorithm, i.e., HERMES. Sections 5 analyzes the HERMES performance. Finally, Section 6 concludes the paper and indicates future directions.

2 RELATED WORK

There have been many works on various TSN topics, including investigation of time-aware shaper mechanisms [4], proposing fault tolerance techniques [12], techniques to tolerate temporary faults in TSN networks with the use of re-transmissions [5], and schedulability analysis of traffic with different TSN features [29], [14]. A recent comprehensive survey [7] presents the status of research within TSN, including schedulability and scheduling problems, safety and security issues, and evaluation models and tools.

Within the context of TT traffic scheduling in TSN networks, the work in [26] present a scheduling algorithm formalized as an ILP while the works in [24] and [16] present a joint routing and scheduling algorithm formalized as an ILP and as a meta-heuristic scheduling approach based on a Genetic Algorithm (GA) approach, respectively. The work in [9] presents an SMT-based scheduler capable of scheduling networks with several TT queues. The work in [10] proposes a GCL synthesis approach based on Greedy Randomized Adaptive Search Procedure (GRASP) meta-heuristic [20], which takes AVB traffic into account, whereas the work in [11] proposes a joint routing and scheduling approach for TT and AVB traffic by means of an integrated heuristic and meta-heuristic strategy. In the latter work, the K-Shortest Path (KSP) method [28] is utilized for routing, and GRASP is used to schedule both TT and AVB at the same time. Moreover, the work in [8] synthesizes a network topology that supports seamless redundant transmission for TT traffic by proposing a greedy heuristic algorithm for joint topology, routing, and scheduling synthesis.

Protocol	Routing	Multi-queuing	Schedule	ZRJ
HLS	Yes	No	per frame	No
MML	Yes	No	per frame	No
BN	Yes	No	per frame	No
CV	Yes	No	per frame	No
MDP	Yes	No	per frame	No
HERMES	No	Yes	per link	Yes

Table 1: Comparison between heuristic schedulers.

The above-mentioned solutions are mostly based on ILP or constraint programming, while some of them exploit the use of meta-heuristics, e.g., GA. However, these solutions normally are highly time-complex, which makes them not scalable. Few works target heuristic solutions with lower time complexity. For instance, the work in [17] proposes a heuristic routing and scheduling algorithm called Heuristic List Scheduler (HLS) that is limited to a single TT queue, while the work in [25] compares 4 heuristic algorithms combining routing and scheduling (Modified Most Loaded Heuristic (MML), Bottleneck Heuristic (BN), Coefficient of Variation Heuristic (CV) [6][13], and Modified Dot Product Heuristic (MDP) [18]), all with scheduling times greater than 100 ms and unable to handle multiple queues. In this work, we propose a heuristic algorithm, called HERMES, with scheduling times lower than 10 ms that uses multiple TT queues to improve schedulability. Moreover, the proposed algorithm provides two modes, one with zero reception jitter and relaxed reception jitter in the receiver end-station, see Section 3 for detailed description of zero and relaxed reception jitter. The zero reception jitter mode is configurable which is helpful for the applications in which the feature is not essential. Table 1 shows the main differences between the heuristic schedulers mentioned above, including HERMES. The features that are analyzed in this comparison include routing, multi-queuing for TT traffic, scheduling process, and support for zero reception jitter (ZRJ).

3 BACKGROUND

TSN end-stations communicate by transmitting Ethernet frames through routes consisting of links and time-sensitive switches. In TSN, Ethernet frames belong to one of the eight possible priorities. The traffic is classified as one of the three available traffic classes, including TT traffic, AVB traffic, and BE traffic, where TT traffic has higher priority than other traffic classes and BE has the lowest priority. Note that several priorities may cover one traffic class, e.g., AVB can consist of classes A, B, and C, each associated with one priority level. A port of a TSN switch supports up to eight FIFO

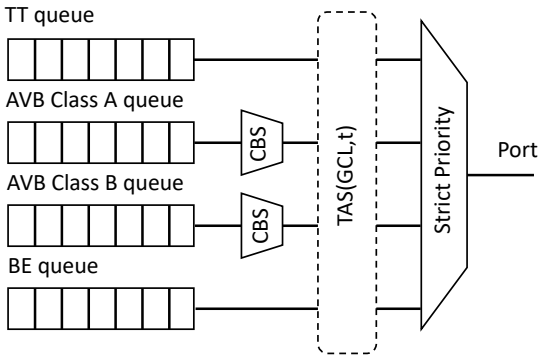


Figure 1: A TSN egress port with four FIFO queues: one TT queue, two AVB queues, and one BE queue.

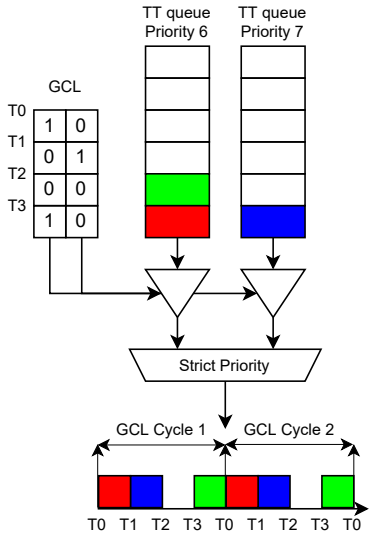


Figure 2: TSN TAS gate mechanism.

queues each of them associated with one priority level. Figure 1 shows an example of a time-sensitive device output port with four queues configured as TT traffic with the highest priority, AVB classes A and B traffic with the medium priority, and a BE traffic class as the lowest priority.

3.1 Time-Triggered Traffic

TT traffic is scheduled offline, which allows to know exactly in which time slot each TT frame is transmitted. This requires that interference between frames must be prevented. This is achieved through the TAS mechanism (see Figure 1). According to this mechanism, each queue has an associated gate that can be open or closed. The frames in a queue can be transmitted when the gate is open, otherwise, the frames are blocked for transmission.The gates are controlled by the GCL, which specifies at which point in time gates should be open, and it is a cyclic list that repeats the schedule. The time that gates are open or closed can be specified at the nanosecond level for each entry of the GCL and we refer to the opening time of a gate as *window*.

Figure 2 shows an example of TAS operation for two TT queues with two different priorities, i.e., priority 6 and 7. In this example, we assume that three TT frames with a period of 4 time units are transmitted through a switch port where one of the TT frames is set to the highest priority 7 (blue frame), while the other two frames (red and green) are set to priority 6. As the periods of the frames are equal, the hyper-period (the least common multiple) of them is 4 time units. Therefore, the GCL cycle is defined as 4 time units allowing gates operation in each time slot and repeating every 4 time units. According to the schedule in this example, which is set in the GCL, at time T0 till T1 the gate for priority 6 queue is open (shown as 1 in GCL), whereas the gate for priority 7 is closed (shown as 0 in GCL) allowing transmission of the red frame. Further, in the time slot between T1 and T2, the blue frame can be transmitted as the gate for priority 7 is open. Between T2 and T3 both gates are closed, thus no transmission can occur, and finally, the gate of priority 6 queue is open in the last time slot that allows the transmission of the last frame, i.e., the green frame. Two cycles of frame transmissions are represented at the bottom of Figure 2.

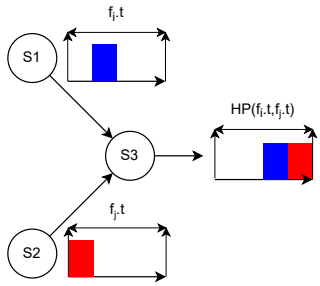
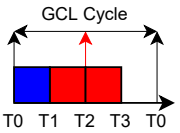


Figure 3: Multi-hop behavior of TSN and GCLs.

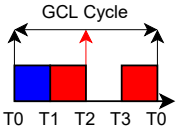
On the other hand, TSN supports multi-hop communication which imposes other restrictions when scheduling. For a frame to be transmitted on a link, it must have been previously transmitted through the preceding links in the route of the frame. Furthermore, the order of frames in transmission is also important. Considering that the queues in the switches are FIFO, if a frame arrives to a switch before another one, it will also be transmitted first. Figure 3 shows an example of a multi-hop schedule. The figure shows three switches (S1, S2, and S3) and three links, two of them connecting S1 and S2 with S3 and one in the S3 output. Two frames are exchanged between these 3 switches, one blue frame is sent by S1 and one red frame is sent by S2 and both frames are forwarded by S3 through the output link. As we can see, both are sent after they are received by S3. However, according to the schedule that is decided for this case (for whatever the reason), the red frame arrives before the blue frame to S3 but the blue frame is sent by S3 before the red frame, thus this schedule would not be possible with a single queue. In this case, the red and blue frames must have been assigned to different queues and hence have different priorities.

Finally, interference-free transmission of frames ensures their zero jitter transmission. This means that the variations in the

transmission and reception of each frame with respect to the schedule will be zero, assuming that the clock drift is zero. However, in this work, not only the jitter but also the reception jitter has been considered. The reception jitter is defined as the variability of the instant of reception by the receiver end-station of a frame with respect to its period. For example, Figure 4 shows the schedule over the GCL cycle of a TSN output port connected to the input port of the receiving end-station. In this schedule two frames are shown, a blue frame with a period of 4 time units and a red frame with a period of 2 time units. According to the schedule example in Figure 4a, the blue frame has zero reception jitter since it is always transmitted at time unit 0, whereas the red frame has reception jitter since it is transmitted at time unit 1 in its first instance (T1) of the hyper-period and at time 0 in its second instance (T2). Zero reception jitter is particularly interesting in heterogeneous systems combining TSN components and legacy components that cannot adopt TSN synchronization mechanisms. In these cases, a frame with zero reception jitter helps the synchronization of the applications even if the devices are not properly synchronized. Throughout the paper, we denote zero reception jitter by ZRJ and reception jitter by RJ. Note that in this case, as shown in Figure 4b, it would be enough to delay the transmission of the second instance of the red frame by one time unit for both frames to have ZRJ.



(a) Schedule of two frames with and without reception jitter.



(b) Schedule of two frames, both with zero reception jitter.

Figure 4: Difference between reception jitter and zero reception jitter.

3.2 AVB and BE Traffic

AVB frames are not scheduled offline, i.e., they are scheduled via CBS once they arrive at the switch port. The gates are normally open for them unless TT traffic is to be transmitted. The CBS aims at improving the Quality-of-Service (QoS) of lower priority traffic while ensuring a minimum of bandwidth utilization. Finally, BE frames have no real-time guarantees, thus they will be transmitted when their gate is open and the CBS is negative for all higher priority traffic.

4 PROPOSED SCHEDULING ALGORITHM

We developed what we call Heuristic multi-queue Scheduler (HERMES) which generates the global schedule for the transmission of TT traffic. Our goal is to reduce the scheduling time while achieving high schedulability through the use of different numbers of TT queues and providing zero jitter. Moreover, HERMES can provide zero reception jitter.

HERMES calculates the GCL of each egress port of the end-stations and TSN switches. To do this, unlike other heuristic algorithms that schedule frame by frame, our algorithm decides the schedule link by link (and then for each link deciding the schedule of each frame to be transmitted in that link) starting with the destination links and ending with the source links scheduling each frame as late as possible according to their timing constraints. The reason why we design HERMES to calculate the schedule link by link instead of frame by frame is that in this way the conflicts between frames are detected before the entire frame has been scheduled. On the contrary, when the network is scheduled frame by frame, the schedule of one frame may hinder the scheduling of the other frames and the algorithm will have to schedule that frame again throughout all its links. On the other hand, links are scheduled from destination to source because the destination link is the most restrictive link especially if the frame has more restrictive reception time constraints such as frames that must be received with zero reception jitter. In addition, in each links, each frame is scheduled as late as possible so that the preceding links have enough time margin between the frame's period start and the offset of the same frame in the last scheduled link. However, this does not imply that the frames will have the highest possible latencies because an improvement as simple as looking for the minimum offset of all frames and moving all frames earlier by that amount can be applied.

The use of multiple queues for TT traffic to improve the schedulability is thoroughly explained in Section 4.2. However, it is worth to mention that HERMES does not consider relative priorities between TT queues, i.e., all queues have the same priority. The isolation of frames by only opening the gate of a single queue at a time eliminates the arbitration among TT frames which is performed by the Strict Priority module (see Figure 1) if more than one queue has its gate open. The queues are only used to help the scheduler meeting the order condition explained in Section 3.

Although HERMES uses only unicast frames, the algorithm would work equally well with multicast frames. As we will see in the algorithm description, HERMES only schedules those links whose frames have already been scheduled in the subsequent links. In this way, in the case of multicast frames, when scheduling the link before the fork, the following links will already be scheduled and therefore the algorithm can continue to function normally only taking into account the offset and restrictions of several following links instead of a single following one.

4.1 System Model

In this work the communication network model consists of two main sets, one for the links $\mathcal{L}$ and one for the TT-frames $\mathcal{F}_{\mathrm{TT}}$. Each link $l \in \mathcal{L}$ is unidirectional, is defined by its identifier, and has a parameter $l.\phi$ indicating in which phase of the execution of HERMES the link will be scheduled. Indeed, the execution of the scheduling algorithm presented in this paper is divided into phases, with a total of Φ phases. In the case of full-duplex links, two links with opposite directions are instantiated. On the other hand, each frame $f \in \mathcal{F}_{\mathrm{TT}}$ is characterized by seven parameters $f = \langle t, w, d, q, u, n, \mathcal{S} \rangle$, i.e.,

(1) the period $f.t$,
(2) the length of the frame or the size of the window needed to transmit the frame $f.w$,
(3) the deadline $f.d$,

(4) the queue of the frame in all egress ports of the whole route $f.q$,
(5) a parameter to decide in which order the frames in the links are scheduled, which in this case is the frame utilization $f.u$ (see Algorithm 2 in Section 4.2),
(6) the number of links in the route $f.n$, and
(7) a set $f.\mathcal{S}$ containing the route and the schedule of each link in the route.

Each element $s \in f.\mathcal{S}$ includes three parameters $s = \langle \zeta, \iota, O \rangle$:

(1) the link $s.\zeta$ of the route assigned in reverse order, i.e. $s_1.\zeta$ being the destination link and $s_{f.n}.\zeta$ the source link,
(2) the number of instances $s.\iota$ of the frame in the link, and
(3) a set $s.O$ indicating the offset of each instance according to the period start of the instance.

4.2 HERMES

The input to the scheduler consists of a list of TT frames characterized by their period, frame length, deadline, and routing expressed as a vector of unidirectional link identifiers. With this list, HERMES executes the following steps to obtain the schedule.

Algorithm 1, DivPhases: First of all, as mentioned before, links are divided into phases where all frames in all links assigned to that phase are scheduled togther. The only condition for a link l_i to be assigned to a phase $l_i.\phi$ is that all frames transmitted through that link $f \in \mathcal{F}_{\text{TT}} : f.s_j.\zeta = l_i$ must have all previous links (links closer to destination) assigned to previous phases $\forall f.s_k.\zeta : k < j | f.s_k.\zeta.\phi < f.s_j.\zeta.\phi$. To do that, if there are links not assigned to any phase, the algorithm adds a new phase and checks if links not assigned are ready to be assigned in the new phase according to previous conditions. Note that, as mentioned before, links in the route are ordered from destination to source, i.e., routes are scheduled backwards and in each phase all links which, fulfilling the above conditions, are independent can be scheduled. Figure 5 presents a network example where the squares with positive numbers represent end-stations and the circles with negative numbers represent switches. For this example, Table 2 shows a list of frames and their corresponding routes expressed as a list of pairs of end-station/switch identifiers indicating the link and its direction. Moreover, Table 3 shows how links in reverse order (from destination to source) are assigned to phases and how these links are delayed (represented by arrows) in the scheduling process until

Algorithm 1: HERMES - DivPhases

```
Data: L, F_TT
Result: ∀l ∈ L return l.φ
procedure DivPhases
    for ∀l ∈ L do l.φ ← NULL end
    Φ ← 1
    while ∃l ∈ L : l.φ = NULL do
        for f_i.s_j, f_k.s_x ∈ F_TT, f.S : f_i.s_{j-1}! =
          NULL ∧ f_k.s_{x-1} = NULL do
            if ∃! f_i.s_j.ζ = f_k.s_x.ζ then
                f_i.s_j.ζ.φ ← Φ
            end
        end
        Φ ← Φ + 1
    end
```

Algorithm 2: HERMES - AssignFrameUtilization

```
Data: F_TT
Result: ∀f ∈ F_TT return f.u
procedure AssignFrameUtilization
    for ∀f ∈ F_TT do
        f.u ← (f.w / f.d) · f.n
    end
```

Algorithm 3: HERMES – Schedule

```
Data: L, F_TT
Result: ∀f ∈ F_TT return f.S
procedure Schedule
    for ∀f ∈ F_TT do f.q ← 1 end
    for p = 1..Φ do
        for ∀l ∈ L : l.φ = p do
            HP ← LCM({f.t : f.s.ζ = l})
            for ∀f.s ∈ f.S : f.s.ζ = l do f.s.ι ← HP / f.t end
            for ∀f ∈ F_TT, f.s_x ∈ f.S : f.s_x.ζ = l from
              highest to lowest f.u do
                for i = f.s_x.ι..1 do
                    f.s_x.o_i ← min(f.s_{x-1}.O, f.d) − f.w
                    while Collision(f.s_x.o_i) ∨
                      Order1(f.s_x.o_i) ∨ Order2(f.s_x.o_i) do
                        if Collision(f.s_x.o_i) then
                            f.s_x.o_i ← f.s_x.o_i − 1
                        else
                            if Order1(f.s_x.o_i) then
                                f.q ← f.q + 1
                                if f.q > Q then
                                    f.s_x.o_i ← f.s_x.o_i − 1
                                end
                            end
                            if Order2(f.s_x.o_i) then
                                f.q ← f.q + 1
                                if f.q > Q then
                                    return Unschedulable
                                end
                            end
                        end
                        if f.s_x.o_i < 0 then
                            return Unschedulable
                        end
                    end
                end
            end
        end
    end
```

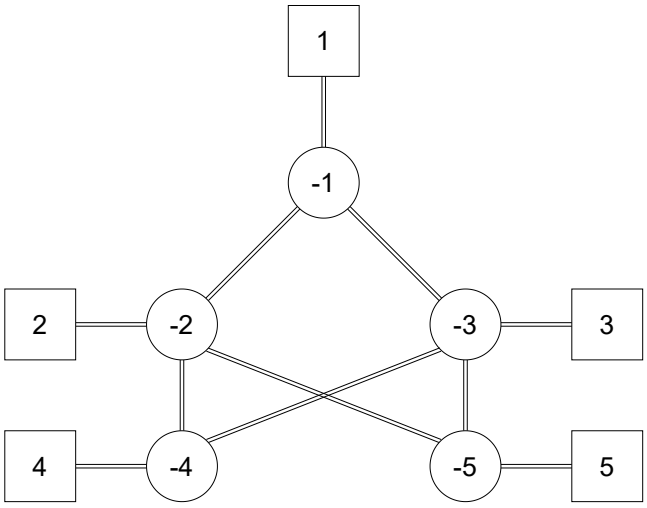


Figure 5: TSN Network example.

the preceding links are assigned to a previous phase. Finally, for this example, Table 4 shows the resulting distribution of the links in phases. The reason for scheduling the links in reverse order is that, due to the need for determinism in the reception link, this link becomes the most restrictive. This is particularly important to obtain a zero reception jitter schedule as mentioned in Section 4. On the other hand, note that, although this scheduler can be used in feed-forward networks, the routes cannot present triple dependencies in a loop. For example, the routes of the frames in Table 5 cannot be distributed in phases. The reason is that, as it can be seen in Table 6, the triple dependency in the loop causes a deadlock in the DivPhases algorithm as some links in the route will be indefinitely delayed.

Frame	Route
f1	1 -1 ; -1 -2 ; -2 2
f2	2 -2 ; -2 -1 ; -1 -3 ; -3 3
f3	2 -2 ; -2 -4 ; -4 4
f4	3 -3 ; -3 -5 ; -5 5
f5	3 -3 ; -3 -1 ; -1 1
f6	3 -3 ; -3 -4 ; -4 -2 ; -2 2
f7	4 -4 ; -4 -2 ; -2 -1 ; -1 1
f8	5 -5 ; -5 -2 ; -2 -1 ; -1 1
f9	5 -5 ; -5 -3 ; -3 3

Table 2: Example of frame routes for Figure 5

Algorithm 2, AssignFrameUtilization: To decide which frames in the link will be scheduled first, the utilization of the frame along its route is calculated according to the formula $f.u = \frac{f.w}{f.d} \cdot f.n$. This formula can change depending on the needs. However, in this case, we sought to prioritize the frames that required the most bandwidth at specific time periods (between the period start and deadline) as this causes the free space for scheduling to be consumed very quickly, so it is important to prioritize them to provide them with the space they need.

Frame	ϕ 1	ϕ 2	ϕ 3	ϕ 4	ϕ 5	ϕ 6
f1	-2 2	-1 -2	1 -1			
f2	-3 3	-1 -3	-2 -1	2 -2		
f3	-4 4	-2 -4	→	2 -2		
f4	-5 5	-3 -5	→	→	→	3 -3
f5	-1 1	-3 -1	→	→	→	3 -3
f6	-2 2	→	→	-4 -2	-3 -4	3 -3
f7	-1 1	→	-2 -1	-4 -2	4 -4	
f8	-1 1	→	-2 -1	-5 -2	5 -5	
f9	-3 3	-5 -3	→	→	5 -5	

Table 3: DivPhases procedure

Phase 1	Phase 2	Phase 3	Phase 4	Phase 5	Phase 6
-1 1	-1 -2	1 -1	2 -2	-3 -4	3 -3
-2 2	-1 -3	-2 -1	-4 -2	4 -4	
-3 3	-2 -4		-5 -2	5 -5	
-4 4	-3 -1				
-5 5	-3 -5				
	-5 -3				

Table 4: Resulting link distribution in phases

Frame	Route
f1	1 -1 ; -1 -3 ; -3 -4 ; -4 4
f2	3 -3 ; -3 -4 ; -4 -2 ; -2 -1 ; -1 1
f3	4 -4 ; -4 -2 ; -2 -1 ; -1 -3 ; -3 3

Table 5: Example of frame routes with feed-forward dependencies for Figure 5

Frame	ϕ 1	ϕ 2	ϕ 3	ϕ 4
f1	-4 4	→	→	...
f2	-1 1	→	→	...
f3	-3 3	→	→	...

Table 6: Infinite DivPhases procedure

Algorithm 3, Schedule (l.2-8): Firstly, we initialize the queues by assigning queue 1 to all frames. Then, HERMES goes through all the phases and, within each phase, all the links, and calculates the hyper-period (HP) by calculating the Least Common Multiple (LCM) of all frames of each link and the number of instances of each frame in the link by dividing the HP by the frame period $f.t$. In this way, HERMES schedules phase by phase, where in each phase the links can be scheduled in parallel because they are independent. Finally, in each link, the schedule of the frames is done in order of descending $f.u$, instance by instance from last ($f.s_x.\iota$) to the first instance in the HP. This is to prevent, in case of having frames with deadlines bigger than periods, later instances to be scheduled before previous instances.

Algorithm 3, Schedule (l.9-10): Secondly, we initialize the offset of each instance of the frame in the link to the minimum between the deadline of the frame and the release of the frame in the previous link (the following link if we consider order from source to destination) if any. Then we check if the offset assigned to the instance of the frame makes it collide with another previously scheduled

instance and if the order of reception and transmission in the switch between this link and the previous one is adequate. **Algorithm 3, Schedule (l.11-34):** Finally, if there is a collision, defined as

$$\begin{aligned} &Collision(f_i.s_j.o_k) = \exists f_m.s_j.o_n : \\ &\left[f_m.s_j.o_n + f_m.t \cdot (n-1) \leq f_i.s_j.o_k + f_i.w + f_i.t \cdot (k-1)\right] \wedge \\ &\left[f_m.s_j.o_n + f_m.w + f_m.t \cdot (n-1) \geq f_i.s_j.o_k + f_i.t \cdot (k-1)\right], \end{aligned} \tag{1}$$

the frame moves backward until it encounters an empty space. Once an empty space is found, the reception and transmission order in the switch is checked. If the frame arrives at the switch later than another frame with which shares the transmission link and which is also transmitted later than the frame under scheduling in the shared transmission link, i.e.,

$$\begin{aligned} &Order1(f_i.s_j.o_k) = \exists f_m.s_j.o_n : \\ &\left[f_m.s_{j-1}.\zeta = f_i.s_{j-1}.\zeta\right] \wedge \\ &\left[f_m.s_j.o_n + f_m.t \cdot (n-1) < f_i.s_j.o_k + f_i.t \cdot (k-1)\right] \wedge \\ &\left[f_m.s_{j-1}.o_{n'} + f_m.t \cdot (n'-1) > f_i.s_{j-1}.o_{k'} + f_i.t \cdot (k'-1)\right] \end{aligned} \tag{2}$$

the switch must change the queue or move backward in the schedule so that it arrives earlier than the frame instance in order conflict. On the other hand, if the frame arrives at the switch earlier than another frame that shares the transmission link and is transmitted earlier than the frame under scheduling in the shared transmission link, i.e.,

$$\begin{aligned} &Order2(f_i.s_j.o_k) = \exists f_m.s_j.o_n : \\ &\left[f_m.s_{j-1}.\zeta = f_i.s_{j-1}.\zeta\right] \wedge \\ &\left[f_m.s_j.o_n + f_m.t \cdot (n-1) > f_i.s_j.o_k + f_i.t \cdot (k-1)\right] \wedge \\ &\left[f_m.s_{j-1}.o_{n'} + f_m.t \cdot (n'-1) < f_i.s_{j-1}.o_{k'} + f_i.t \cdot (k'-1)\right] \end{aligned} \tag{3}$$

the only option is to change the queue or the network configuration will be unschedulable for the ordering approach used. Moreover, if the offset becomes negative due to the frame advance, the configuration will also be unschedulable. Note that changing the queue is not enough, it is also necessary to check that such a change does not affect the order conditions of the previously scheduled links but for the sake of simplicity, this has not been included in the algorithm.

On the other hand, in HERMES frames can be configured as RJ or ZRJ. For the sake of simplicity, this is not reflected in the algorithm but it consists of forcing all offsets to be equal on the reception link of the frames configured as ZRJ. If a frame is configured as RJ, it can be received by the receiver at different points in time even if reception is deterministic. For example, if one frame is scheduled as RJ and has a period of 4s, HERMES may schedule it in a loop of 3 instances where the first instance has an offset of 1s, the second instance has an offset of 3s, and the third instance has an offset of 2s. In this way, the instances of this frame will be received at seconds 1, 7, 10, 13, 19, and so on. However, if a frame is configured as ZRJ, it will be received by the receiver at a constant rate equal to the period. For example, if one frame is scheduled as ZRJ and has a period of 4s, HERMES schedules it in a way that the offsets of all instances are the same. In this way, if the frame has an offset of 2s the frame will be received at seconds 2, 6, 10, 14, 18, and so on. As mentioned before, this behavior is especially interesting for legacy devices that cannot implement TSN synchronization protocols but want to execute applications in a TT fashion, or want to exchange their own legacy synchronization frame with other legacy devices through TSN.

5 EVALUATION OF HERMES

5.1 Experimental setup

For the evaluation of HERMES, in this paper, we used the LETRA evaluation toolset (ETS) developed in [15]. LETRA ETS provides a set of integrated tools capable of performing automated experiments regarding the scheduling and schedulability analysis of TSN networks. In this section, we will explain LETRA ETS and the modifications that have been done for this paper. The reader is referred to [15] for more information about LETRA ETS.

The toolchain of the LETRA ETS used in this work is depicted in Figure 6. The main input to the ETS is the network configuration, including the network topology and traffic characteristics. For this paper, we use two network topologies both following a line-star topology. The network topologies are depicted in Figure 7 and they are a small single-switch network consisting of 3 nodes (S1) and a larger three-switch network consisting of 9 nodes (S3).

For the traffic characteristics, we set the traffic to be only TT as we exclude the effect of HERMES on lower priority traffic in this stage. The network bandwidth is set to 10Mbps to prevent generating too many frames when reaching loads around 90% utilization to avoid taking more than a week to conduct each round of experiments due to the CP scheduler and the network generator which are the two most time-consuming tools. The maximum number of generated frames is set to 100, however, depending on the selected utilization, the frame number can be different. The frame length is selected within the range [500,1000] Bytes. The minimum and maximum allowed periods are set to $200\mu s$ and $1000\mu s$ respectively, while deadlines were assigned the same values as the corresponding periods. Note that the frames will be generated such that the utilization of all links will be the one selected as an input, e.g., when we select 10% utilization the traffic generator selects the frame sizes and routes to obtain 10% on all links if possible.

The first step of LETRA ETS is generating random traffic. We used the network configuration as input of the network generator to generate 1700 sets of TT traffic randomly for each network topology (100 TT traffic configurations for each of the values of the utilization, which are taken [from 10% to 90% in steps of 5% of utilization]).

The next step is LETRA, as it can be seen in Figure 6, which is a mapping tool to map the generated traffic into TSN traffic classes, i.e., TT, AVB and BE. In this paper, we are only interested in TT traffic, thus, we skip the traffic mapping. However, ETS is integrated in a way that the output of each tool is the input of the next. For this reason, we used LETRA only as a translator between the output of the network generator and the input of the schedulers.

Finally, each generated network processed by LETRA is used as input to both HERMES and a CP scheduler [21]. The CP scheduler runs in the only available mode which is with one queue and RJ while HERMES, depending on the experiment, is configured with up to 4 TT queues and with the frames configured in either mode RJ or ZRJ.

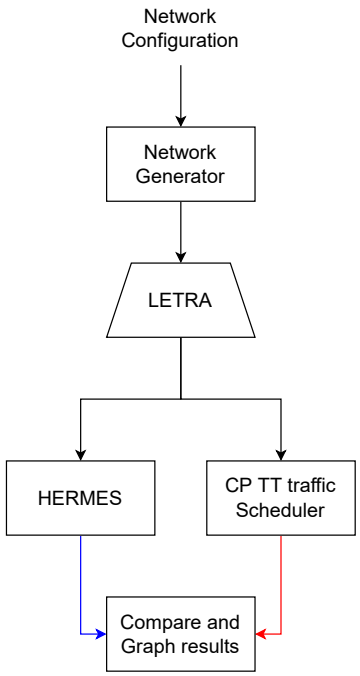
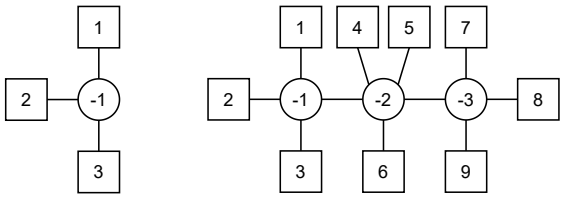


Figure 6: LETRA evaluation tool set modification.



(a) Single-switch network (S1). (b) Three-switches network (S3).

Figure 7: Experimental network topologies.

Once the scheduling of all the generated networks is done, we compare the two schedulers, i.e., HERMES and CP, with respect to the time that it takes for each of them to give a solution and the number of networks for which each of the schedulers is able to find a schedule (number of networks considered as schedulable by each scheduler). The experiments are done for different values of the network utilization, which is the same on all network links, e.g., 10% utilization is considered in all links of the network. We show the results for different network utilization in several graphs in the following sections. In the graphs that show the number of schedulable networks, the circles indicate the percentage of networks generated that could be scheduled while error bars represent the error in the percentage obtained through the binomial analysis with 95% certainty. Additionally, these graphs include dashed trend lines obtained through logistic regression adjustment to present the trend of changes. Moreover, Table 7 compares the schedulability between HERMES in modes RJ and ZRJ and the CP scheduler on networks S1 and S3. In the table, we analyze schedulability $S(u)$, as a function of the utilization u, for the range of utilizations under analysis ($u \in [10\%, 90\%]$). Since the schedulability $S(u)$ is sampled, we approximate it by a logistic regression, that we indicate with the notation $\widehat{S}(u)$. Then we define the accumulated schedulability for a network x as:

$$AS_x = \int_{0.1}^{0.9} \widehat{S}_x(u)du \tag{4}$$

which we use to compare schedulers using the accumulated scheduling ratio defined as:

$$ASR_{x,y}[\%] = \frac{AS_x}{AS_y} \cdot 100 \tag{5}$$

to measure the percentage of schedulable networks of x compared to y. On the other hand, in the graphs that show the time taken by each scheduler to give a solution, the circles show the average time needed to get the schedule in milliseconds for each utilization level, while error bars are calculated using the gamma distribution with 95% certainty. Moreover, the graphs include a dashed trend line obtained by fitting the data to an exponential function or a polynomial function of order 1 or 2.

5.2 Results of the scheduling time

We start with the evaluation by analyzing the time it takes to give a schedule for both network topologies. In Figs. 8 and 9 scheduling times for all HERMES modes and the number of queues as well as the scheduling time of the CP scheduler for networks S1 and S3 respectively are shown. In both graphs, we can see how the scheduling time of the CP scheduler is exponentially increasing with the percentage of utilization and the number of frames while HERMES remains with scheduling times below 10ms. This implies that HERMES exhibits scheduling times from tens of times lower than the CP scheduler to thousands of times lower for high utilization values. Furthermore, we can see how for 50% utilization the scheduling time in the S1 network for the CP scheduler is 3000ms while for the S3 network it is 8000ms, which also shows a large increase in scheduling time with the size of the network.

On the other hand, Figs. 10 and 11 show a detail of the scheduling times specifically for HERMES in RJ mode for networks S1 and S3 respectively. Both graphs show an increase in scheduling time proportional to the square of the utilization, which is related to the number of frames. On the other hand, the scheduling time is proportional to the longest path between two end-stations. In this case, as shown in Figure 7, the ratio between the longest routes is 4/2. For a utilization of 60%, we observe that in the S1 network HERMES with 2 and 3 queues takes 1 and 2 ms respectively while in the S3 network for the same number of queues the scheduling time is 2 and 4 ms, which corresponds to the ratio calculated above. Finally, it is also possible to identify that scheduling time doubles with every extra queue, for example, for a 60% utilization in the S3 network, HERMES takes 2, 4, and 8 ms to get a schedule with 2, 3, and 4 queues respectively. Although the complexity doubles with each extra queue used, the fact that the queues are limited to 8 reduces its impact and allows HERMES to remain scalable.

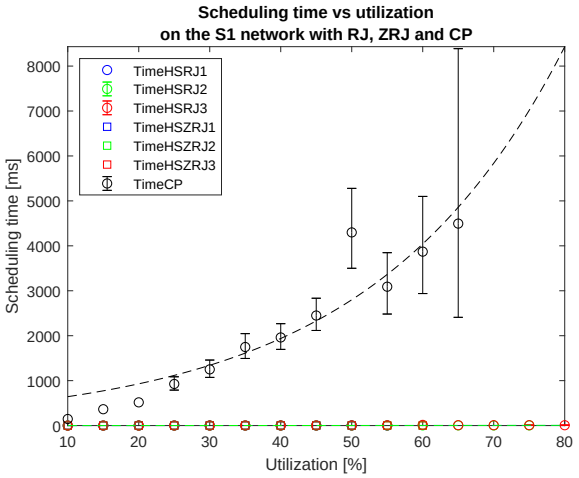


Figure 8: Scheduling time for different levels of network utilization on network S1 of a CP scheduler and HERMES with and without zero reception jitter with 1, 2 and 3 queues.

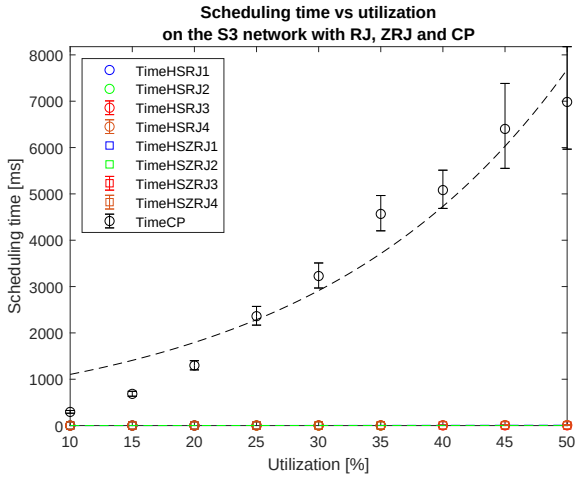


Figure 9: Scheduling time for different levels of network utilization on network S3 of a CP scheduler and HERMES with and without zero reception jitter with 1, 2, 3 and 4 queues.

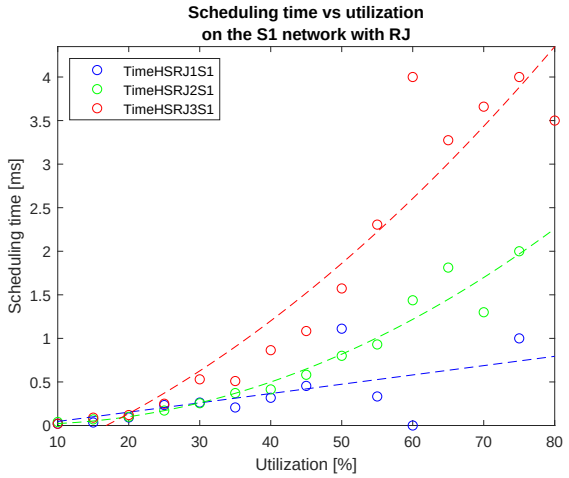


Figure 10: Scheduling time for different levels of network utilization on network S1 of HERMES with reception jitter with 1, 2 and 3 queues.

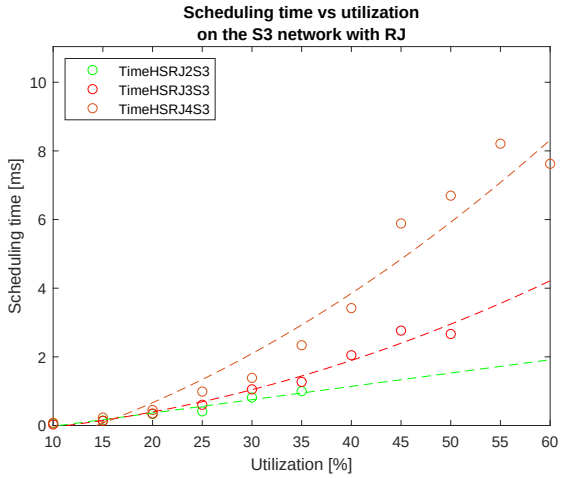


Figure 11: Scheduling time for different levels of network utilization on network S3 of HERMES with reception jitter with 2, 3 and 4 queues.

5.3 Results of the schedulability

Figure 12 shows in black the schedulability of the CP scheduler for a single-switch network (S1) with the set of generated networks. On the other hand, in blue, green, and red we can see the HERMES schedulability with 1, 2, and 3 TT queues respectively in mode RJ (with jitter in the reception). The first observation is that it would be enough to increase the number of queues to 2 to obtain the same schedulability as the CP scheduler with 1 queue but, in addition, with three queues it is even possible to exceed by more than 32% the schedulability achieved by the CP scheduler, as shown in the first column of Table 7. These results, together with those shown in the previous subsection, show the usefulness of HERMES in contexts where the number of queues is not a constraint but the schedulability time is, e.g., run-time configurations.

Figure 13 shows the HERMES schedulability in zero reception jitter (ZRJ) mode for the S1 network. This graph shows that in this case, from 2 queues onwards, the schedulability stagnates due to the tough constraint that the ZRJ mode imposes. However, in the second column of Table 7 it can be seen how the schedulability is lower than in RJ mode, except for the case of one queue where the ZRJ mode restriction facilitates the scheduling of certain cases. Since, in general, the schedulability in ZRJ mode is lower than the schedulability in RJ mode, it is recommended to limit this mode to frames that really require it.

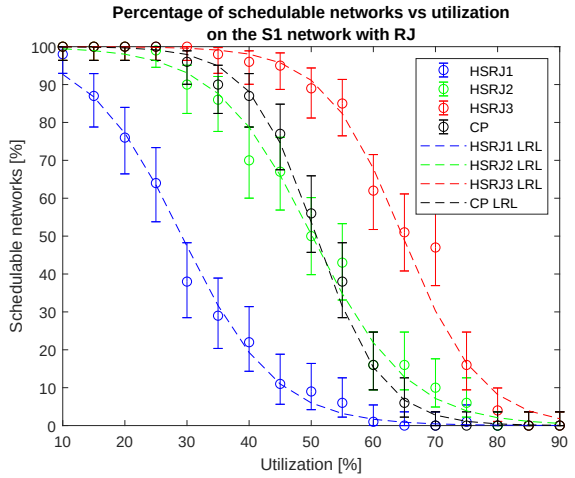


Figure 12: Schedulability for different levels of network utilization on network S1 of a CP scheduler and HERMES allowing reception jitter with 1, 2 and 3 queues.

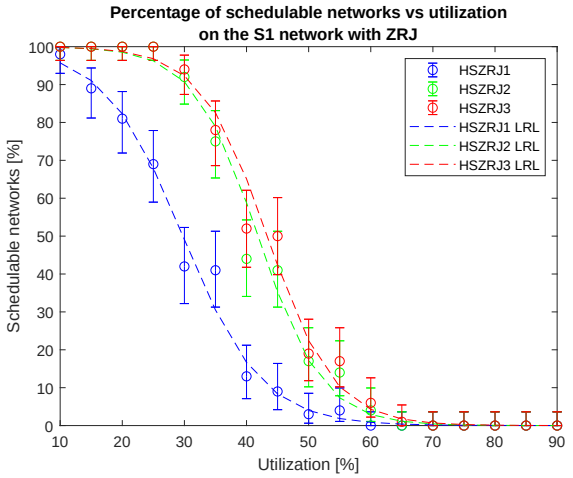


Figure 13: Schedulability for different levels of network utilization on network S1 of HERMES in zero reception jitter mode with 1, 2 and 3 queues.

Figure 14 shows in black the schedulability of the CP scheduler for a three-switches network (S3) with the set of generated networks. On the other hand, in blue, green, red, and orange we can see the HERMES schedulability with 1, 2, 3, and 4 TT queues respectively in mode RJ. In this graph, we can see how HERMES scales worse the longer the route of the frame, being necessary up to 4 queues to surpass the schedulability levels that are attainable with the CP scheduler, as shown in the third column of Table 7. However, it can also be noticed that with 3 queues 81% schedulability is achieved so the improvement in scheduling time can still be worthwhile. For example, different approaches can be tried to order the frames, apart from frame utilization, so that, although each has a lower schedulability, together can cover all the cases covered by the CP scheduler even in less time since different frame scheduling orders in the links will provide different schedules.

Figure 15 shows the HERMES schedulability in ZRJ mode for the S3 network. Similar to what happened in the S1 network, in this case, the schedulability in ZRJ mode also stagnates. However, the stall occurs after the third queue. On the other hand, as we can see in the fourth column of Table 7, with fewer queues the ZRJ constrain may slightly improve schedulability but for more queues, the difference is greater and increasing so, since more queues are

	Network S1		Network S3	
N° Q	$ASR_{RJ,CP}$	$ASR_{ZRJ,RJ}$	$ASR_{ZRJ,CP}$	$ASR_{ZRJ,RJ}$
1	51.04	101.58	17.37	102.59
2	98.50	80.54	55.09	104.62
3	131.99	62.73	81.59	96.33
4	–	–	101.35	81.98

Table 7: Schedulability comparison between HERMES with different number of queues (Q) in mode RJ and ZRJ and the CP scheduler on networks S1 and S3.

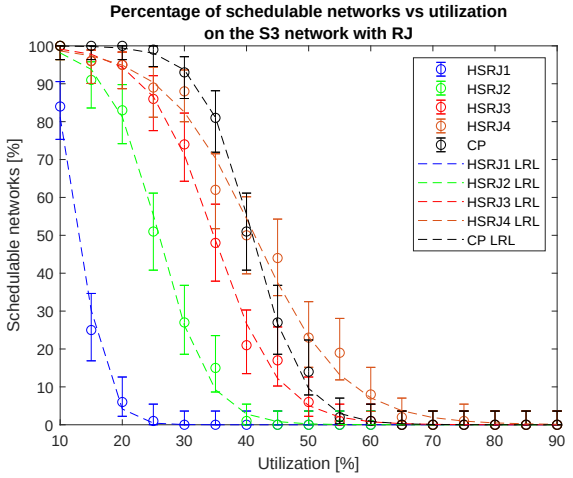


Figure 14: Schedulability for different levels of network utilization on network S3 of a CP scheduler and HERMES allowing reception jitter with 1, 2, 3 and 4 queues.

needed to achieve high values of schedulability for this kind of traffic, again, this mode should be left for very specific frames if high schedulability wants to be achieved.

6 CONCLUSION AND FUTURE WORK

We argued that developing fast scheduling algorithms are crucial specially for adaptive and evolutionary systems. Therefore, in this work, we have developed a fast heuristic scheduler for TT traffic in TSN networks called HERMES that can match the level of schedulability of reference schedulers by using several TT queues. We also use the LETRA ETS to evaluate HERMES performance showing that by using several queues HERMES can outperform the schedulability of CP schedulers with a single queue but with

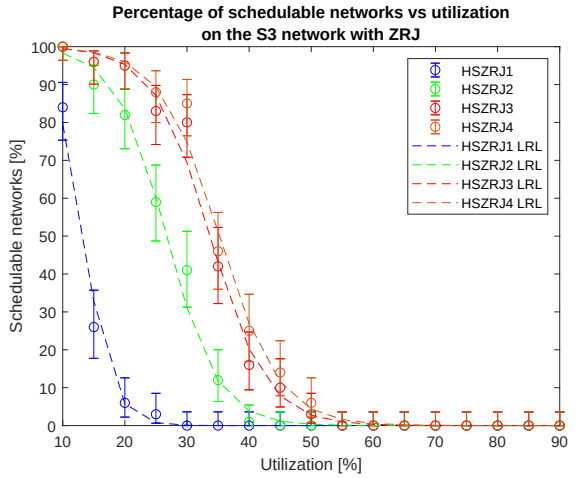


Figure 15: Schedulability for different levels of network utilization on network S3 of HERMES in zero reception jitter mode with 1, 2, 3 and 4 queues.

HERMES exhibiting scheduling times of less than 10 ms, which implies that HERMES is hundreds or thousands of times faster. In addition, HERMES supports the integrative capability of TSN by providing a more restrictive ZRJ mode that facilitates the integration into TSN networks of legacy devices that cannot implement TSN's own synchronization mechanisms.

In this work, we focus on TT traffic scheduling. However, in previous works, we have developed a TSN mapping tool and an AVB analyzer. Therefore, the next step is to integrate all these tools to create a toolset capable of mapping and scheduling traffic taking into account the real-time requirements of all kinds of traffic.

ACKNOWLEDGMENTS

This research is supported by the Swedish Knowledge Foundation (KKS) under the FIESTA project, the Swedish Governmental Agency for Innovation Systems (VINNOVA) under the DESTINE and PROVIDENT projects.

REFERENCES

[1] 2010. IEEE Standard for Local and Metropolitan Area Networks - Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams. *IEEE Std 802.1Qav-2009 (Amendment to IEEE Std 802.1Q-2005)* (2010), C1–72. https://doi.org/10.1109/IEEESTD.2009.5375704

[2] 2010. IEEE Standard for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks Amendment 14: Stream Reservation Protocol (SRP). *IEEE Std 802.1Qat-2010 (Revision of IEEE Std 802.1Q-2005)* (2010). https://doi.org/10.1109/IEEESTD.2010.5594972

[3] 2020. IEEE Standard for Local and Metropolitan Area Networks–Timing and Synchronization for Time-Sensitive Applications. *IEEE Std 802.1AS-2020 (Revision of IEEE Std 802.1AS-2011)* (2020), 1–421. https://doi.org/10.1109/IEEESTD.2020.9121845

[4] G. Alderisi, G. Patti, and L. Lo Bello. 2013. Introducing support for scheduled traffic over IEEE audio video bridging networks. In *Conf. Emerging Technologies Factory Automation*.

[5] Inés Álvarez, Ignasi Furió, Julián Proenza, and Manuel Barranco. 2021. Design and Experimental Evaluation of the Proactive Transmission of Replicated Frames Mechanism over Time-Sensitive Networking. *Sensors* 21, 3 (2021), 756.

[6] Emmanuel Arzuaga and David R Kaeli. 2010. Quantifying load imbalance on virtualized enterprise servers. In *Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering*. 235–242.

[7] Mohammad Ashjaei, Lucia Lo Bello, Masoud Daneshtalab, Gaetano Patti, Sergio Saponara, and Saad Mubeen. 2021. Time-Sensitive Networking in Automotive Embedded Systems: State of the Art and Research Opportunities. *Journal of Systems Architecture* 110 (September 2021), 1–47.

[8] Ayman A Atallah, Ghaith Bany Hamad, and Otmane Ait Mohamed. 2018. Fault-resilient topology planning and traffic configuration for IEEE 802.1 Qbv TSN networks. In *2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS)*. IEEE, 151–156.

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

[10] Voica Gavriluț and Paul Pop. 2018. Scheduling in time sensitive networks (TSN) for mixed-criticality industrial applications. In *14th IEEE International Workshop on Factory Communication Systems (WFCS)*. IEEE, 1–4.

[11] Voica Gavriluț, Luxi Zhao, Michael L Raagaard, and Paul Pop. 2018. AVB-aware routing and scheduling of time-triggered traffic for TSN. *IEEE Access* 6 (2018), 75229–75243.

[12] S. Kehrer, O. Kleineberg, and D. Heffernan. 2014. A comparison of fault-tolerance concepts for IEEE 802.1 Time Sensitive Networks (TSN). In *IEEE Emerging Technology and Factory Automation*.

[13] Leonard Kleinrock. 1975. Queueing systems, volume i: Theory, vol. I.

[14] Lucia Lo Bello, Mohammad Ashjaei, Gaetano Patti, and Moris Behnam. 2020. Schedulability analysis of Time-Sensitive Networks with scheduled traffic and preemption support. *Journal of Parallel and Distributed Computing*, 144, (2020).

[15] Daniel Bujosa Mateu, Mohammad Ashjaei, Alessandro V Papadopoulos, Julian Proenza, and Thomas Nolte. 2021. LETRA: Mapping Legacy Ethernet-Based Traffic into TSN Traffic Classes. In *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 1–8.

[16] Maryam Pahlevan and Roman Obermaisser. 2018. Genetic algorithm for scheduling time-triggered traffic in time-sensitive networks. In *2018 IEEE 23rd international conference on emerging technologies and factory automation (ETFA)*, Vol. 1. IEEE, 337–344.

[17] Maryam Pahlevan, Nadra Tabassam, and Roman Obermaisser. 2019. Heuristic list scheduler for time triggered traffic in time sensitive networks. *ACM Sigbed Review* 16, 1 (2019), 15–20.

[18] Rina Panigrahy, Kunal Talwar, Lincoln Uyeda, and Udi Wieder. 2011. Heuristics for vector bin packing. *research. microsoft. com* (2011).

[19] Michael Lander Raagaard and Paul Pop. 2017. Optimization algorithms for the scheduling of IEEE 802.1 Time-Sensitive Networking (TSN). *Tech. Univ. Denmark, Lyngby, Denmark, Tech. Rep* (2017).

[20] Mauricio GC Resende and Celso C Ribeiro. 2014. GRASP: Greedy randomized adaptive search procedures. In *Search methodologies*. Springer, 287–312.

[21] Niklas Reusch, Silviu S Craciunas, and Paul Pop. 2021. Dependability-Aware Routing and Scheduling for Time-Sensitive Networking. *arXiv preprint arXiv:2109.05883* (2021).

[22] R. Salazar, T. Godfrey, L. Winkel, N. Finn, C. Powell, B. Rolfe, and M. Seewald. 2018. *Utility Applications of Time Sensitive Networking White Paper (D3)*. Technical Report. IEEE. https://mentor.ieee.org/802.24/dcn/18/24-18-0022-03-sgtg-utility-applications-of-time-sensitive-networking-white-paper-d3.docx

[23] S. Samii and H. Zinner. 2018. Level 5 by Layer 2: Time-Sensitive Networking for Autonomous Vehicles. *IEEE Communications Standards Magazine* 2, 2 (2018), 62–68. https://doi.org/10.1109/MCOMSTD.2018.1700079

[24] Eike Schweissguth, Dirk Timmermann, Helge Parzyjegla, Peter Danielis, and Gero Mühl. 2020. ILP-based routing and scheduling of multicast realtime traffic in time-sensitive networks. In *2020 IEEE 26th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. IEEE, 1–11.

[25] Ammad Ali Syed, Serkan Ayaz, Tim Leinmüller, and Madhu Chandra. 2021. Dynamic scheduling and routing for TSN based in-vehicle networks. In *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 1–6.

[26] Marek Vlk, Kateřina Brejchová, Zdeněk Hanzálek, and Siyu Tang. 2022. Large-scale periodic scheduling in time-sensitive networks. *Computers & Operations Research* 137 (2022), 105512.

[27] M. Wollschlaeger, T. Sauter, and J. Jasperneite. 2017. The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0. *IEEE Industrial Electronics Magazine* 11, 1 (2017), 17–27. https://doi.org/10.1109/MIE.2017.2649104

[28] Jin Y Yen. 1971. Finding the k shortest loopless paths in a network. *management Science* 17, 11 (1971), 712–716.

[29] Luxi Zhao, Paul Pop, Zhong Zheng, and Qiao Li. 2018. Timing analysis of AVB traffic in TSN networks using network calculus. In *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. 25–36. https://doi.org/10.1109/RTAS.2018.00009