Development of an Ethernet-Based Heuristic Time-Sensitive Networking Scheduling Algorithm for Real-Time In-Vehicle Data Transmission

Hyeong-Jun Kim 1,*, Min-Hee Choi 1, Mah-Ho Kim 2 and Suk Lee 1

1 School of Mechanical Engineering, Pusan National University, Busan 46241, Korea; june423@pusan.ac.kr (H.-J.K.); minhee303@pusan.ac.kr (M.-H.C.)
2 Division of Automotive Engineering, Dong-eui Institute of Technology, Busan 47230, Korea; mhkim@dit.ac.kr
* Correspondence: slee@pusan.ac.kr; Tel.: +82-051-510-3091

Abstract: The rapid development and adaptation of advanced driver assistance systems (ADAS) and autonomous driving increases the burden of in-vehicle networks. In-vehicle networks are now required to provide a fast data rate and bounded delay for real-time operation, while conventional protocols such as controller area networks, local interconnected networks, and FlexRay begin to show limitations. Ethernet-based time-sensitive network (TSN) technology has been proposed as an alternative. TSN is a set of Ethernet standards being developed by the IEEE TSN task group, which aims to satisfy requirements such as real-time operation, stability, and low and bounded latency, and it can be used in automotive, industrial, and aerospace applications. This study introduces several standards for Ethernet traffic scheduling based on TSN technology and proposes a heuristic-based scheduling algorithm for Ethernet scheduling. In addition, three network configurations are simulated using OMNeT++ to show the applicability. The heuristic TSN scheduling algorithm is a straightforward and systematic procedure for practical network designers.

Keywords: automotive Ethernet; time-sensitive networking (TSN); audio video bridge (AVB); in-vehicle network; Ethernet scheduling

1. Introduction

In-vehicle network (IVN) protocols such as controller area network (CAN), local interconnect network (LIN), and FlexRay have been employed to accommodate various requirements for data exchange between electronic control units (ECUs) [1–3]. However, existing IVN protocols are showing their technical limitations as vehicles tend to be smarter by employing advanced driver assistance systems (ADAS), vehicle-to-everything (V2X), and autonomous driving. In particular, sensors used for autonomous driving such as light detection and ranging (LiDAR), radar, and cameras require a large amount of data exchange for sensor fusion and artificial intelligence processing [4–10].

CAN, LIN, and FlexRay, which are traditional networks for vehicles, have limitations with respect to rapidly transferring increased amount of vehicle information. IEEE 802.3 Ethernet-based network technology has been proposed as an alternative [11]. Ethernet is a local area network and communication technology developed in 1973, which was standardized to IEEE 802.3 in 1983; it is one of the most widely used communication methods. Ethernet provides a high transmission rate and excellent universality due to international standardization and supports various protocols such as the transmission control protocol/internet protocol (TCP/IP) and scalable service-oriented middleware over IP (SOME/IP). However, in a traditional Ethernet, a contention method is used to allow multiple nodes to access a communication network, but this cannot be applied to vehicles due to the limitations in real-time data processing, stability, and bounded latency. However, a recently developed Ethernet that adopts a switching technology is
able to overcome these limitations, and some work has been undertaken to investigate its application to communication for vehicles and factory automation [11]. In particular, problems associated with wire weight and high costs have been solved by adopting 100/1000BASE-T1 (Open Alliance BroadR-Reach) as a hardware interface used by the IEEE. In addition, standardization has been actively implemented to improve real-time processing and stability, and to meet the high specifications required for use in vehicles.

Audio video bridging/time-sensitive networking (AVB/TSN) is a set of standards in the Ethernet layer 2 of the open systems interconnection (OSI) model, which aims to ensure bounded latency and deterministic characteristics for transmission of data that are time-sensitive in Ethernet networks, and to improve the quality of service (QoS) that manages traffic and bandwidth according to the importance of the application [12]. In particular, TSN is a set of standards to ensure critical traffic transmission for control of various devices and systems and represents an extension of AVB developed for audio and video synchronization. In addition, TSN includes synchronization technology standards for devices in Ethernet-based networks, scheduling for meeting real-time requirements, stream reservation protocols, and frame preemption, for application to automotive, industrial, and aerospace fields. Typical standards used to configure TSNs include IEEE 802.1Qbv, IEEE 802.1AS, and IEEE 802.1Qcc [13].

In particular, IEEE 802.1Qbv includes an Ethernet scheduling mechanism and parameters based on the time-aware shaper (TAS), which can guarantee the deterministic characteristics of Ethernet-based networks [14]. IEEE 802.1Qbv includes a mechanism for controlling eight queues in an egress port through gate opening and closing and specifies the required parameters. However, this standard does not provide a method for creating tables for frame scheduling. Various scheduling methods can be used for each TSN-applied network, and the approach to scheduling and the network environment may affect the network performance of deterministic systems.

Multiple studies have proposed network scheduling optimization methods for vehicles, such as time-triggered (TT) Ethernet and FlexRay. Steiner applied the satisfiability modulo theories (SMT) solver to ensure TT traffic transmission in TT-Ethernet, and then introduced a slot between schedules to reduce the rate-constrained (RC) traffic delay in a follow-up study [15,16]. Craciunas formalized a scheduling problem using logical constraints and proposed an optimization method based on SMT and mixed integer programming (MIP) [17]. Selicean et al. proposed a scheduling method for TT traffic to minimize the worst-case delay (WCD) using a tabu search-based metaheuristic [18]. Most studies deal with TT traffic scheduling, which is fundamentally different from scheduling methods for TSN-scheduled traffic (ST). A TSN does not directly schedule TT traffic, but instead schedules a gate control list (GCL) for data transmission according to the gate state that controls a queue. A number of researchers proposed a scheduling method for IEEE 802.1Qbv. Gavrilu et al. proposed a greedy randomized adaptive search procedure (GRASP)-based method to schedule both TSN and AVB traffic in industrial applications but did not measure the reduction in traffic communication delay [19]. Dürr defined the concept of “critical data scheduling” as the “no-wait job-shop scheduling problem”, and proposed a method based on a tabu search [20]. However, the end-to-end delay of ST was not considered, although an algorithm to compress scheduling, thereby reducing bandwidth waste due to the guard band, was proposed. Joung proposed a regulating scheduler based on deficit round robin (DRR) without separating traffic classes in TSN network [21].

This study proposes a heuristic scheduling algorithm for ST on a TSN Ethernet network used as an in-vehicle network, assuming that the network traffic is light to moderate as most in-vehicle networks are designed. The proposed algorithm is intended for automotive electronics engineers to come up with an acceptable schedule without too much mathematical formulation and iteration. The heuristic TSN scheduling algorithm includes a method for assigning STs on links in the route from their source to destination and an adjustment procedure for reducing end-to-end delay. In addition, this study verifies
the applicability of the proposed heuristic TSN scheduling algorithm based on several scenarios along with their simulations.

The present paper is organized into five sections. In Section 2, standards related to TSN scheduling are introduced, and, in Section 3, the proposed heuristic TSN scheduling algorithm is described. In Section 4, the performance evaluation environment of the proposed scheduling method and the performance evaluation results are discussed. Lastly, in Section 5, the conclusions of the study are presented.

2. Time-Sensitive Networking (TSN)

2.1. Traffic Scheduling with TSN

A TSN comprises a set of standards for OSI layer 2 that is standardized by the IEEE; it specifies various functions, such as synchronization between devices that employ Ethernet-based networks, traffic scheduling, network management, redundancy, resource management, and stream reservation. For example, IEEE 802.1Qbv specifies the mechanism of TAS-based traffic scheduling, and IEEE 802.1Qbu uses the preemption technique to reduce the guard band size of ST. IEEE 802.1Qcc allows for effective scheduling by acquiring all relevant information through network integration management.

2.2. Time-Aware Shaper

Scheduled traffic (ST) is traffic transmitted with high priority according to a schedule table, which should not be interrupted by rate-constrained (RC) and best-effort (BE) traffic. Here, RC refers to traffic transmitted according to the credit that varies with time used for waiting and transmission. BE traffic utilizes the available bandwidth to the largest extent possible without any traffic control function, such that traffic transmission is not guaranteed [22]. Traffic to be transmitted through the TAS is stored in eight queues located in the switch transmission port, according to priority, and transmitted according to the gate state. Figure 1 shows an example for the TAS-based scheduling mechanism, where traffic flows according to gate control list (GCL) value of the gate driver [23]. The GCL has eight digits, each of which corresponds with the queue. If the GCL value is “0”, the corresponding gate is closed, and, if the value is “1”, the gate is open. The gate driver performs open and close operations repeatedly in specific time intervals according to the synchronized time. The queue on the left in Figure 1 is an ST queue, in which ST is transmitted when the gate is opened while the RC and BE traffic gates are closed. In contrast, the gate of the ST queue is closed when other gates aside from the ST queue are open for RC and BE traffic. The eight gates are controlled according to the GCL; once the list is completed, the system returns to the beginning and iteration begins again. ST transmission can be guaranteed by suppressing the transmission of other class traffic.

Figure 1. Example time-aware shaper (TAS)-based scheduling process.
Figure 2 shows the data transmission process of a combined credit-based shaper (CBS) and TAS system, which transmits RC traffic [14]. Here, the CBS for RC transmission has a communication delay of up to 2 ms based on seven hops; however, it does not meet the requirement for vehicles of a maximum delay of 100 µs based on five hops [24]. With the CBS only, even if critical traffic flows in, it will be queued until the transmission currently in progress is complete; if transmission of a large amount of traffic, such as audio or video data, is underway, a significant delay to critical traffic transmission may occur. In contrast, TAS generates a schedule in advance to transmit the ST, which is accomplished within an mandatorily determined time period. In the system shown in Figure 2, transmission of RC and BE traffic is impossible until the ST is transmitted, and the RC and BE traffic is transmitted according to the CBS algorithm in the section outside the ST transmission section. This method allows for the transmission of the real-time and deterministic traffic information required for vehicles.

In order to prevent RC and BE traffic from spilling over the ST time slot, IEEE 802.1 Qbv has a guard band with a maximum frame size (MFS) of 1500 bytes before each ST time slot during which no transmission is allowed. This can guarantee the start of ST transmission and the bounded latency. However, the presence of a guard band will cause bandwidth loss within the network. To overcome this shortcoming, IEEE 802.1Qbu specifies a preemption mechanism. It allows minimizing the size of the guard band and interrupting ongoing transmission of RC and BE traffic by ST traffic. In addition, IEEE 802.1Qcc specifies the protocol, procedures, and management approach for network resource configuration in time-sensitive applications. This improves the network resource management performance over that of IEEE 802.1Qat, as well as the interface between the user and network components. Moreover, three models can be used for TSN configuration: fully distributed, centralized network/distributed user, and fully centralized models. Centralized network configuration (CNC), which configures network resources in place of TSN applications, and centralized user configuration (CUC), which exchanges information with the CNC to configure the TSN instead of an end station, may be needed according to the TSN configuration. In particular, CNC controls the network bridge of a TSN and determines the flow route and scheduling, as well as the configuration of bridges and switches. To achieve this, the CNC exchanges the requirements of network designs through an additional protocol with the CUC, and it generates a route and schedule that are sent to the bridges and switches.

With IEEE 802.1Qcc, there are only three models available for TSN configuration; although requirements and protocols for configuration are proposed, network designers are responsible for implementing route setup and scheduling. The present study assumes...
that there is a CNC that can configure all switches in the route that ST passes through in the network. In Section 3, a heuristic TSN scheduling algorithm is described.

3. Heuristic TSN Scheduling Algorithm

Although IEEE 802.1Qbv specifies the mechanism for communicating information of scheduling in a TSN, the designer is responsible for coming up with a schedule. This study proposes a heuristic scheduling algorithm that assumes that switches within a network can share information on the period, size, and priority of all STs. The STs are assigned on the schedule table, which eventually results in Gate Control Lists (GCLs) of all switches in the network. The proposed algorithm begins with selecting the ST with the highest priority and then allocates time slots, of which the length is proportional to the length of the ST, for all links on the path from the source node to the destination node. This process is repeated for other STs with descending order of priority. After all the STs have been selected and assigned, the second phase of scheduling tries to reduce end-to-end delay by adjusting assignments on the links.

3.1. Heuristic TSN Scheduling Algorithm

The proposed algorithm is based on some assumptions. The first assumption is that the traffic is always schedulable. With heavy traffic, it may not be possible to schedule all the STs, which is not the case for this study. The second assumption is that delay caused by the propagation of an electromagnetic wave is so short that we can ignore the delay from the transmitter to the receiver. We also assume that the delay inside a switch is negligible when the outgoing link is free.

Figure 3 shows the heuristic TSN scheduling algorithm used for creating the schedule table for a TSN network. This algorithm calculates $T_{all}$, which is the least common multiple of all ST transmission periods. $T_i$ denotes the period of the i-th ST. ST index i is given such that the i-th ST has higher or equal priority than the $(i+1)$-th ST. For each ST$i$, the aim is to determine how many times ST$i$ should be transmitted during the entire cycle $T_{all}$, which is denoted by $N_{STi}$. The indices $j$ and $k$ used in Figure 3 are the number of ST$i$s that have been placed on the schedule table and the number of links that ST$i$ has passed on the link from source node to destination node, respectively. Scheduling begins with assigning ST$i$ to the earliest available time slot required for the first link ($L_{i1}$). The time slot’s length is proportional to the length of ST$i$, i.e., total number of bits of ST$i$ divided by data rate. For ST$i$, this process repeats for all the links to the destination, i.e., until $k = h_i$ where $h_i$ is the number of links ST$i$ must pass. Then, one ST$i$ is assigned from source to destination node, and this repeats as many times as the ST$i$ must be transmitted during the $T_{all}$ by increasing $j$ by one until $j = N_{STi}$. When the allocation for all ST$i$ is completed ($i = n$), the next phase of scheduling is to reduce end-to-end delay of STs if possible. The end-to-end delay is defined by the time interval from the start of the first bit from the source to the arrival of the last bit at the destination. Because low-priority STs are allocated after STs with high priority, their time slots are more likely to be delayed, which increases the end-to-end delay. If that is the case, we may delay the start from the source because we know it will be delayed in route to the destination, which eventually reduces the end-to-end delay. Therefore, we start with ST$n$ that has the lowest priority, and we check whether the end-to-end delay can be reduced by delaying start time of the selected ST. If end-to-end delay can be reduced, we reallocate time slots for the ST along its route. Otherwise, the schedule remains the same. This process is repeated for the next lowest priority by decrementing $i$. This scheduling algorithm can be easily understood using the example in the next subsection.

3.2. An Example

Figure 4 shows a network to explain the heuristic TSN scheduling algorithm shown in Figure 3. The network consists of six nodes (end systems, ES) and two switches (SW), which exchange six STs at 100 Mbps full duplex. Table 1 shows the source, destination,
priority, size, and period of each ST in the network. Following the flowchart in Figure 3, we can easily see that $T_{all} = 2000$ and ST$_1$ should appear four times in the schedule.

**Figure 3.** Structure of the heuristic time-sensitive networking (TSN) scheduling algorithm using the branch and bound technique.

**Figure 4.** Example network for the heuristic TSN scheduling method.
Table 1. Information on scheduled traffic (ST) transmitted via the network.

| [Source, Destination] | Priority | Size (bytes) | Period (µs) |
|------------------------|----------|--------------|-------------|
| ST1 [ES1, ES6]         | 6        | 125          | 500         |
| ST2 [ES1, ES5]         | 4        | 250          | 1000        |
| ST3 [ES2, ES5]         | 7        | 125          | 2000        |
| ST4 [ES2, ES4]         | 5        | 250          | 500         |
| ST5 [ES3, ES4]         | 2        | 125          | 1000        |
| ST6 [ES3, ES6]         | 3        | 250          | 2000        |

Figure 5a shows the first part of the initial schedule. The schedule includes all the necessary links to send STs, depicted by the pairs of links in brackets, e.g., [ES1, SW1] and [SW1, SW2]. The horizontal axis is time. This schedule is obtained by allocating STs to the links that lead to their destination. According to the scheduling algorithm explained earlier, we begin with STs with the highest priority. Therefore, we take ST3 first and assign the links from ES2 to ES5, i.e., [ES2, SW1], [SW1, SW2], and [SW2, ES5]. The link capacity occupied by ST3 is represented by a block with the length corresponding to the time required to transmit all the bits (125 × 8 bits / 100 Mbps = 10 µs for ST3). This assignment is repeated three more times at 500, 1000, and 1500 µs, which is not shown in the figure. Next, ST1 with the second highest priority is assigned its links required to reach the destination. We can see that ST1 is delayed at SW1 because the link [SW1, SW2] is already assigned for ST3. By repeating this process, we can come up with the initial schedule in Figure 5a.

Figure 5b is obtained by adjusting the initial schedule following the steps described in the right-hand side of Figure 3. In this part, we start with the ST with the lowest priority, i.e., ST5. When inspecting the link capacity allocated for ST5 in the initial schedule (Figure 5a), we can see that ST5 is delayed at SW1 because the last link [SW1, ES4] is occupied by ST4, which results in 40 µs end-to-end delay for ST5. With further inspection, we can reduce the end-to-end delay by 10 µs by shifting ST5 to the right on [ES3, SW2] and [SW2, SW1]. Then, we can repeat the process for ST6 by shifting ST6 to the right on [ES3, SW2] to reduce the delay from 60 µs to 50 µs. We can now stop this procedure because the remaining ST delays cannot be improved any further.

Figure 5. Worst-case delay in an example network using the heuristic TSN scheduling method. (a) before re-schedule; (b) after re-schedule
the delay from 60 µs to 50 µs. We can now stop this procedure because the remaining ST delays cannot be improved any further.

4. Realistic Network and Simulation Results

Figure 6 shows a more realistic network to demonstrate the applicability of the proposed scheduling method. The network was configured on the basis of the Society of Automotive Engineers (SAE) International “Benchmark” network and message database [25], with some modifications reflecting current trends in in-vehicle Ethernet implementation. The network consisted of seven nodes and five switches for six STs. We considered three cases as shown in Table 2, one with identical length and period for all STs, another with different lengths and identical period for all STs, the last with different lengths and periods. We used the same transmission speed of 100 Mbps full duplex.

Table 2. Details of the STs used in the simulation.

| Case | [Source, Destination] | Priority | Size (bytes) | Period (µs) |
|------|-----------------------|----------|--------------|-------------|
| 1    | ST1 [ES1, ES7]        | 6        | 125          | 500         |
|      | ST2 [ES1, ES6]        | 5        | 125          | 500         |
|      | ST3 [ES2, ES7]        | 3        | 125          | 500         |
|      | ST4 [ES3, ES6]        | 7        | 125          | 500         |
|      | ST5 [ES3, ES5]        | 4        | 125          | 500         |
|      | ST6 [ES4, ES5]        | 2        | 125          | 500         |
| 2    | ST1 [ES1, ES7]        | 6        | 250          | 500         |
|      | ST2 [ES1, ES6]        | 5        | 125          | 500         |
|      | ST3 [ES2, ES7]        | 3        | 250          | 500         |
|      | ST4 [ES3, ES6]        | 7        | 125          | 500         |
|      | ST5 [ES3, ES5]        | 4        | 250          | 500         |
|      | ST6 [ES4, ES5]        | 2        | 125          | 500         |
| 3    | ST1 [ES1, ES7]        | 6        | 250          | 1000        |
|      | ST2 [ES1, ES6]        | 5        | 125          | 500         |
|      | ST3 [ES2, ES7]        | 3        | 250          | 1000        |
|      | ST4 [ES3, ES6]        | 7        | 125          | 500         |
|      | ST5 [ES3, ES5]        | 4        | 250          | 2000        |
|      | ST6 [ES4, ES5]        | 2        | 125          | 2000        |

Figure 7a shows the results achieved for Case 3 by applying the priority-based assignment only. STs were assigned their respective links in the order of ST4, 1, 2, 5, 3, and 6. Then, in the reverse order, possibility of reducing end-to-end delay was checked, and, if possible, assignments were shifted to reduce the delay, which resulted in the schedule shown in Figure 7b. These two schedules were simulated using the OMNeT++ network simulator, more specifically, the CoRE4INET framework implementing the AVB/TSN function [26].
Figure 7. Worst-case delay in Case 3 using the heuristic TSN scheduling method: (a) before re-schedule; (b) after re-schedule.

Figure 8 shows the end-to-end delays of six STs for three cases. The dark bars represent the delay before the adjustments, while the shaded bars represent the delay after the adjustments. Figure 8a is for Case 1, which shows that the end-to-end delays of ST3, ST5, and ST6 with lower priorities were decreased by the adjustments. Figure 8b,c show delay reduction in ST2, ST3, ST5, and ST6 for Cases 2 and 3, respectively.

Figure 8. End-to-end delay simulation results for the priority-only scheduling method and the heuristic TSN scheduling algorithm: (a) Case 1; (b) Case 2; (c) Case 3.
5. Conclusions

This paper proposed a heuristic TSN scheduling algorithm for in-vehicle TSN networks. In particular, STs are assigned in the order of their priorities, and then assignments are adjusted to decrease end-to-end delays. In addition, the proposed algorithm was applied to several network scenarios along with simulations for end-to-end delays. With these results, the proposed scheduling algorithm demonstrated its applicability for real-world TSNs under certain assumptions. In addition, this algorithm offers a straightforward and systematic procedure for IVN engineers.

It should be noted that this study only focused on the end-to-end delay of ST and did not evaluate the number of guard bands generated when ST was transmitted. Further study is, thus, required because scheduling performance can be very sensitive to the number of guard bands under medium-to-heavy traffic. Additional studies are also needed for considering other classes of traffic and all types of network delay for scheduling performance.

Author Contributions: Conceptualization, H.-J.K.; Formal analysis M.-H.C.; Software, M.-H.C.; Supervision, M.-H.K.; Visualization, M.-H.K.; Writing—original draft, H.-J.K.; Writing—review and editing, S.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the System Industrial Strategic Technology Development Program (10079961, Development of a deterministic DCU platform with less than 1 µs synchronization for autonomous driving system control) funded by the Ministry of Trade, Industry, and Energy (MOTIE, Korea).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hu, Q.; Luo, F. Review of secure communication approaches for in-vehicle network. Int. J. Automot. Technol. 2018, 19, 879–894. [CrossRef]
2. Darbandi, A.; Yoon, S.H.; Kim, M.K. Schedule construction under precedence constraints in flexray in-vehicle networks. Int. J. Automot. Technol. 2017, 18, 671–683. [CrossRef]
3. Park, I.; Sunwoo, M. FlexRay network parameter optimization method for automotive applications. IEEE Trans. Ind. Electron. 2011, 58, 1449–1459. [CrossRef]
4. Uhlemann, E. Introducing connected vehicles. IEEE Veh. Technol. Mag. 2015, 10, 23–31.
5. Doecke, S.; Grant, A.; Anderson, R.W. The real-world safety potential of connected vehicle. Traffic Inj. Prev. 2015, 16, 531–535. [PubMed]
6. Kim, M.H.; Lee, S.; Ha, K.N.; Lee, K.C. Implementation of a fuzzy-inference-based low-speed, close-range collision warning system for the urban area. Proc. Inst. Mech. Eng. Part D J. Automob. Eng. 2013, 227, 234–245. [CrossRef]
7. Lee, S.H.; Lee, S.; Kim, M.H. Development of a driving behavior-based collision warning system using a neural network. Int. J. Automot. Technol. 2018, 19, 837–844. [CrossRef]
8. Zolock, J.; Senatore, C.; Yee, R.; Larson, R.; Curry, B. The Use of Stationary Object Radar Sensor Data from Advanced Driver Assistance Systems (ADAS) in Accident Reconstruction. SAE Tech. Pap. 2016. [CrossRef]
9. Park, T.; Samii, S.; Shin, K.G. Design optimization of frame preemption in real-time switched ethernet. In Proceedings of the 2019 Design Automation and Test in Europe Conference and Exhibition, Florence, Italy, 25–29 March 2019.
10. Liu, X.; Nie, Z.; Li, D.; Yu, H. Design of an improved Ethernet AVB model for real-time communication in in-vehicle network. In Proceedings of the 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference 2019, Chengdu, China, 15–17 March 2019.
11. Lee, K.C.; Lee, S. Performance evaluation of switched Ethernet for real-time industrial communications. Comput. Stand. Interfaces 2002, 24, 411–423. [CrossRef]
12. IEEE std. IEEE Standard for a Transport Protocol for Time-Sensitive Applications in Bridged Local Area Networks; IEEE-SA: New York, NY, USA, 2016.
13. Bello, L.L.; Steiner, W. A perspective on IEEE time-sensitive networking for industrial communication and automation systems. Proc. IEEE 2019, 107, 1094–1120. [CrossRef]
14. IEEE std. 802.1 Qbv-Enhancements for Scheduled Traffic; IEEE-SA: New York, NY, USA, 2016.
15. Steiner, W. An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks. In Proceedings of the 2010 31st IEEE Real-Time Systems Symposium, San Diego, CA, USA, 30 November–3 December 2010; pp. 375–384.
16. Steiner, W. Synthesis of static communication schedules for mixed-criticality systems. In Proceedings of the International Symposium on Object Component Service-Oriented Real-Time Distributed Computing Workshops, Newport Beach, CA, USA, 28–31 March 2011; pp. 11–18.
17. Craciunas, S.S.; Oliver, R.S. Combined task- and network-level scheduling for distributed time-triggered systems. *Real Time Syst.* **2016**, *52*, 161–200. [CrossRef]

18. Tamaș-Selicean, D.; Pop, P.; Steiner, W. Design optimization of TTEthernet-based distributed real-time systems. *Real Time Syst.* **2014**, *51*, 1–35. [CrossRef]

19. Gavrilut, V.; Pop, P. Scheduling in time sensitive networks (TSN) for mixed-criticality industrial applications. In Proceedings of the 2018 14th IEEE Int. Workshop on Factory Communication Systems, Imperia, Italy, 13–15 June 2018.

20. Dürr, F.; Nayak, N.G. No-wait packet scheduling for IEEE time-sensitive networks (TSN). In Proceedings of the International Conference on Real-Time Networks and Systems, Brest, France, 19–21 October 2016.

21. Joung, J. Regulating Scheduler (RSC): A Novel Solution for IEEE 802.1 Time Sensitive Network (TSN). *Electronics* **2019**, *8*, 189. [CrossRef]

22. Steiner, W. TTEthernet dataflow concept. In Proceedings of the 8th IEEE International Symposium Network Computing Application, Cambridge, CA, USA, 9–11 July 2009.

23. Meyer, P. Extending IEEE 802.1 AVB with time-triggered scheduling: A simulation study of the coexistence of synchronous and asynchronous traffic. In Proceedings of the IEEE Vehicular Networking Conference, Boston, MA, USA, 16–18 December 2013.

24. Pannell, D. AVB-Generation 2 Latency Improvement Options. Available online: http://www.ieee802.org/1/files/pub-lic/docs2011/new-pannell-latency-options-0311-v1.pdf (accessed on 26 October 2020).

25. Tindell, K.; Burns, A. Guaranteeing message latencies on controller area network (CAN). In Proceedings of the 1st International Can Conference, Geneva, Switzerland, 25–27 May 1994; pp. 1–11.

26. Falk, F.; Hellamns, D.; Carabelli, B.; Nayak, N.; Dürr, F.; Kehrer, S.; Rothermel, K. NeSTiNg: Simulating IEEE time-sensitive networking (TSN) in OMNeT++. In Proceedings of the International Conference on Network and System, Munich, Germany, 18–21 March 2019.