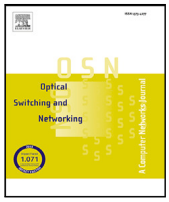
Optical Switching and Networking 50 (2023) 100753

Contents lists available at ScienceDirect

Optical Switching and Networking

journal homepage: www.elsevier.com/locate/osn



A zone-based optical intra-vehicle backbone network architecture with dynamic slot scheduling

Onur Alparslan *, Shin'ichi Arakawa, Masayuki Murata

Graduate School of Information Science and Technology, Osaka University, 1-5 Yamadaoka, Suita, 565-0871, Osaka, Japan

ARTICLE INFO

Keywords:
Intra-vehicle networks
Optical network
TSN ethernet

ABSTRACT

As Ethernet has a large bandwidth capacity, it is commonly proposed as a backbone for future intra-vehicle networks. However, satisfying the severe hardware reliability requirements of intra-vehicle networks while providing high-bandwidth and low latency by Ethernet may be costly. As a solution, we propose a novel optical intra-vehicle backbone network architecture that may have a lower cost and higher reliability in terms of hardware when compared to Ethernet. However, unlike traditional optical Ethernet architectures, only a single master node has transmitter laser diodes in the backbone of our architecture, so the gateway nodes cannot generate and send packets to the backbone links directly. As the gateways cannot inform the master node and request a slot when they have a new packet to send, a slot scheduling algorithm with polling is necessary to detect and transfer the new packets in the gateways, which may cause higher transmission delays compared to Ethernet. In this paper, we present our optical intra-vehicle backbone network architecture and propose two slot scheduling algorithms. We show that using a dynamic slot scheduling algorithm decreases packet delays when compared to fixed periodic slot scheduling in our architecture. We also evaluate the total delays including traffic shaping and processing delays in an optical TSN Ethernet backbone architecture as a reference. We show that the extra delays due to slot scheduling in our architecture may be negligibly low when compared with traffic shaping and processing delays.

1. Introduction

The automotive industry is one of the world's largest industries by revenue. There is a fierce competition among many automotive companies to increase their market share. The competition is driven by innovation. Recently, the innovation in the automotive industry is mostly based on providing new and better services and features in vehicles. Vehicle manufacturers add new features to the vehicles by mounting embedded systems called electronic control unit (ECU). However, the complexity and requirements of vehicles increase by adding new ECUs. For example, ECUs of some features like self-driving systems and driver-assistance systems (ADAS) require carrying large amount of data with low latency and strict Quality of service (QoS) between multiple ECUs and sensors in intra-vehicle networks [1]. Moreover, adding new multimedia services and faster wireless Internet services to the vehicles further increases the bandwidth requirements in intra-vehicle networks [2,3].

There are several bus architectures (e.g., CAN, LIN, MOST, FlexRay) that were specially designed for intra-vehicle networks. While most of these bus architectures can provide low latency, they cannot satisfy the high bandwidth requirements of recent applications like self-driving systems and multimedia services. Ethernet is commonly proposed as a candidate for the next generation intra-vehicle backbone networks due to its high data transfer rates. However, the traditional Ethernet cannot satisfy the QoS requirements of some services like self-driving systems. As a solution, a set of standards called Time-Sensitive Networking (TSN), which is under development by the Time-Sensitive Networking task group of the IEEE 802.1 working group [4], is commonly proposed for intra-vehicle networks [5]. Another problem is that vehicles operate in harsh environments. To satisfy the high bandwidth and reliability requirements in harsh environments, an optical Ethernet standard for vehicles, which is called OMEGA, is being standardized by an IEEE task force [6]. However, satisfying the severe hardware reliability requirements of intra-vehicle networks while providing high-bandwidth and low latency by using an optical Ethernet may bring a high cost due to the shorter lifetime of laser diodes (LD) at high temperatures in vehicles [7]. The cost is a big concern in the automotive industry because even a small change in the costs can cause a big change in profits. Reducing the costs makes it possible to sell at a lower price, which can bring a big advantage in the competition.

* Corresponding author.
E-mail addresses: a-onur@ist.osaka-u.ac.jp (O. Alparslan), arakawa@ist.osaka-u.ac.jp (S. Arakawa), murata@ist.osaka-u.ac.jp (M. Murata).

https://doi.org/10.1016/j.osn.2023.100753
Received 5 April 2023; Received in revised form 2 June 2023; Accepted 11 June 2023
Available online 14 June 2023

Besides the bus architecture, the network topology is another important factor in the performance and the cost of intra-vehicle networks. Initially, a gateway-based architecture was commonly used in intra-vehicle networks. In this architecture, there were a few number of gateways that were used for only data switching between ECUs. Recently, a domain-based topology, where related ECUs and sensors are connected to the same domain controller, is being used. While a domain-based architecture can decrease the ECU and sensor costs by controlling the related ECUs and sensors by a domain controller, it increases the wiring cost and weight when related ECUs are far away from each other. Nowadays, the automotive industry is planning to shift to a zonal-based architecture, where there are zone gateways at different portions of the vehicle, and ECUs are connected to nearby zonal gateways based on the spatial distance [8]. The zonal-based architecture can greatly decrease the wiring cost and weight. Moreover, it can further optimize the ECU and gateway hardware costs by centralizing the processing in the vehicle [9,10]. On the other hand, an intra-vehicle architecture with a high bandwidth capacity, low latency, and good QoS support is necessary for centralizing the processing in the zone-based architecture.

In this paper, we propose a novel optical intra-vehicle backbone network architecture that may have a lower hardware cost than OMEGA optical Ethernet. We call it Si-based In-vehicle Photonic Network (SiPhON). Its optical hardware was presented in [7,11]. Unlike traditional optical Ethernet architectures, where each gateway is equipped with LDs to transmit data, our architecture uses LDs only in a single master node, which can decrease the cost caused by the failure of fragile LDs in harsh automotive environments. However, using LDs only in the master node imposes an important limitation in our architecture. The gateway nodes cannot generate a signal when they want to send a packet to the backbone. The optical signal for the packet should be generated in the master node and sent to the gateway in a slot. After receiving the signal, the gateway injects the packet into the slot by modulating the optical signal in the slot. Moreover, when there is a new packet to send from a gateway to the master node, the gateway node may not be able to inform the master node immediately to get a slot. The gateway can send more slot requests by only writing the requests to the header of another slot assigned to the same gateway. If there is no slot scheduled for the gateway, the gateway cannot inform the master node. Due to these limitations, the packet transmission delays in our architecture may be higher than Ethernet. To minimize the waiting times of packets for slots, our architecture requires a unique slot scheduling algorithm. In [12], we used a fixed and periodic slot schedule list to show the basic operation of our architecture. The slots were assigned to the gateways in a manually set fixed and periodic pattern, which required the traffic matrix to be known and fixed. Moreover, the fixed and periodic scheduling can cause high packet latencies with bursty traffic.

This paper is an extended version of [12]. The extension in this paper is that we present a new dynamic slot scheduling algorithm that does not require traffic matrix information or manually setting a slot scheduling pattern. By a computer-based simulation study, we show that the dynamic slot scheduling algorithm gives lower end-to-end packet latencies than the fixed and periodic slot scheduling.

The advantage of our SiPhON architecture is that it may have lower hardware costs and higher reliability than Ethernet. We do not claim the new architecture to have superiority over TSN Ethernet in terms of packet delays. On the contrary, our architecture may have higher packet delays than TSN Ethernet due to the lack of LDs in the gateway nodes as explained above. However, our architecture may have a better performance/cost ratio than Ethernet if it causes a small increase in delays while having a much lower cost and higher reliability. To show the scale of extra delays due to slot scheduling in our architecture, we also simulated an optical TSN Ethernet architecture and measured the overall delays including the traffic shaping and processing. We showed that the extra packet delays due to slot scheduling in SiPhON may be negligibly low when compared with the total delays including traffic shaping and processing delays in TSN Ethernet.

As the devices in our architecture have not been fully implemented yet, the processing delays in the architecture are currently unknown. Moreover, there are many traffic shaping and admission control algorithms in the literature so adapting them to our architecture and evaluating their delays requires separate work. Therefore, the evaluation of the processing, traffic shaping, and admission control delays in our architecture are left as a future work.

The remainder of this paper is organized as follows. Section 2 introduces related works in the literature. Section 3 presents our network architecture. Section 4 introduces our new dynamic slot scheduling algorithm and compares it with fixed and periodic slot scheduling. Section 5 describes the simulation scenario, settings and presents the simulation results. Section 6 describes the conclusions and future works.

2. Related work

2.1. Intra-vehicle architectures

Due to the cost and bandwidth limitations of different bus architectures, hybrid combinations of different bus architectures started to be used in vehicles. This brought the necessity of using multilevel topologies with gateways to allow communication between ECUs on different bus architectures. The intra-vehicle architectures with multilevel topologies and multiple bus architectures can be grouped into three generations. The first generation vehicles used a gateway-based architecture, where ECUs were connected to each other and a centralized gateway by grouping based on the underlying bus architecture like CAN, LIN, etc. There may be multiple buses with the same architecture where each bus is dedicated to a group of ECUs with related features or a group of ECUs with close spatial distance. The gateways are responsible for only translation and switching of data between different buses as shown in Fig. 1. While this gateway architecture is a simple solution for using multiple bus architectures, it has important limitations. When there are many buses connected to the same gateway by a star topology, the gateway requires a fast switching fabric with many input/output ports, which increases the cost. Moreover, when there are many ECUs in the vehicle, it becomes difficult to configure and manage each of them. Furthermore, many ECUs have overlapping functionalities and sensors, which unnecessarily increases the cost. It would be better to manage the ECUs and process the sensor data by centralized controllers to prevent overlapping functionalities. However, the gateways do not have a control over the ECUs. Moreover, when there are many ECUs, connecting each bus to a single gateway may require long wiring, which increases both the cost and the weight of the vehicle. The wires are the third heaviest element of a typical vehicle after the engine and chassis [13].

Recently, the vehicle manufacturers shifted to a second generation architecture called domain-based architecture [14] as shown in Fig. 1. In domain-based architecture, there are multiple domain controllers, and each of them is dedicated to a specific feature. The ECUs are grouped based on their functionalities and the related ECUs are connected to the same domain controller. Unlike the gateway architecture, where the gateways are only switching devices, the domain controllers both switch the data and manage the related ECUs connected to them. Moreover, they can carry out the processing of data instead of processing in the ECUs. They can collect and fuse data from multiple ECUs so that they can give better decisions than processing by individual ECUs. Furthermore, it becomes no longer necessary to put a high-speed processor to ECUs for processing. Overlapping functionalities and sensors in ECUs can be avoided. As a result, the domain-based architecture can greatly decrease the cost and complexity of vehicles compared to the gateway architecture.

While the domain-based architecture decreases the cost of ECUs, it still has the wiring cost problem, because the related ECUs may be far

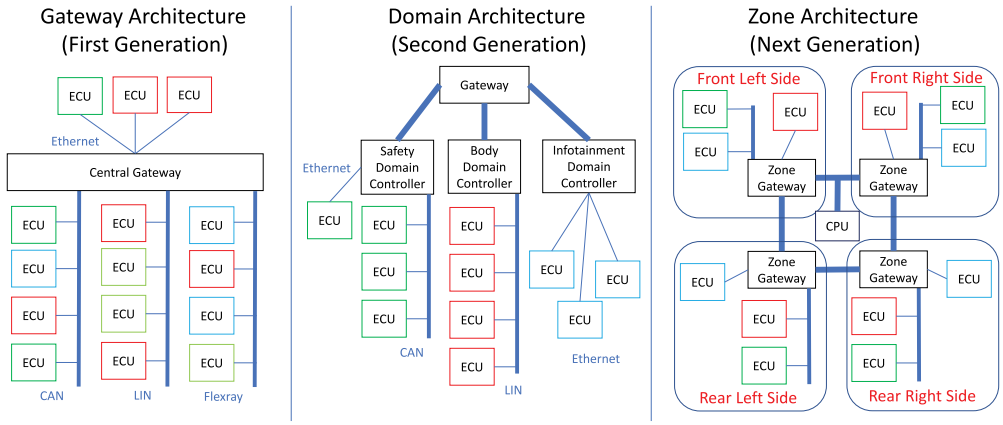


Fig. 1. The evolution of intra-vehicle architectures. Gateway architecture, domain architecture and zone architecture.

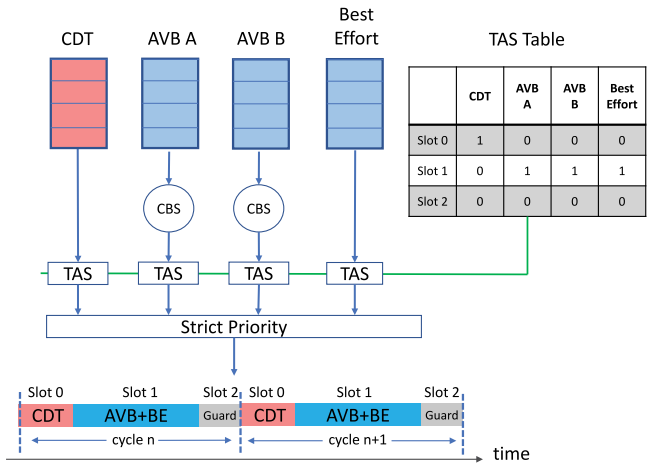
away from each other. For example, sensors like LIDAR, video CAM, etc. are usually far away from each other because they are placed at the edges of the car to maximize the angle of view. Wiring to each edge from many domain controllers may end up requiring very long wires. To solve these problems, a third generation architecture called zone-based architecture is being proposed by the automotive industry [14]. In the zone-based architecture, the vehicle is segmented into topological zones and a zone gateway is assigned to each zone. The ECUs are grouped and connected to the closest zone gateways based on the distance instead of their features. An example zone architecture with four zones and four zone gateways is shown in Fig. 1. The ECUs are connected to the closest zone gateway based on their distance, independent of their bus architecture or functionality. Therefore, the wiring is greatly reduced when compared to a domain-based architecture. In domain-based architecture, the domain controllers can do the data processing of the ECUs directly connected to the domain controller. In zone-based architecture, few number of centralized processors in the vehicle carry out the data processing of ECUs that are at different zones of the vehicle. The processors do not have to be in the same zone as the ECUs. The zone gateways can collect the data from many ECUs and forward them to centralized processors over the intra-vehicle backbone. Therefore, the zone-based architecture may require less number of processors than domain-based architecture, which can further decrease the costs. On the other hand, the zone-based architecture requires an intra-vehicle architecture with a high bandwidth capacity and low latency for centralizing the processing. As our intra-vehicle architecture can satisfy these requirements, we used a zone-based architecture to benefit from its advantages.

2.2. TSN and AVB Ethernet

In the early days of vehicle electronics, the first ECUs were used for simple functions such as fuel injection. The ECUs were connected to sensors by direct wires and there was no connection between the ECUs. As the number of ECUs increased, communication between the ECUs became necessary for providing more advanced features and decreasing the wiring cost. The first car with a communication bus called D-Bus was introduced by BMW in 1987. The communication bus was used for engine control. In 1992, Daimler introduced the first car with CAN bus, which was developed by Bosch in 80 s. Today, CAN is the most common bus architecture in vehicles [15]. After CAN, new bus architectures emerged in a short time. As the cost of implementing CAN in each sensor and ECU was high, Local Interconnect Network (LIN) bus, which is a cheaper and slower bus architecture than CAN, was developed in the 90 s. Moreover, the bandwidth limitations of CAN made it necessary to develop faster bus topologies like Media Oriented Systems Transport (MOST) and FlexRay.

Recent automotive features like self-driving systems require collecting large amounts of data with low latency. Moreover, in the next generation zone-based architecture the data may be processed by centralized processors instead of the ECUs, so the bandwidth capacity, latency and QoS support become even more crucial. While traditional intra-vehicle bus architectures have good QoS capabilities, their link speeds are no longer enough. As Ethernet has already reached very high link speeds, Ethernet is commonly proposed as a strong candidate for the backbone of automotive networks. While the original Ethernet was a best-effort datagram service, Ethernet was been extended with IEEE 802.1p and later 802.1Q [16] to provide some basic QoS capabilities. However, the QoS capabilities in initial 802.1Q standards were far away from satisfying the strict QoS requirements of critical services like safety and driver assist functions in automotive networks. In the 2000s, several new standards were introduced for further extending the QoS capabilities of Ethernet. Audio Video Bridging Task Group was established to prepare the specifications that will allow time-synchronized low latency audio/video streaming services over Ethernet. The task group published a set of technical standards called Audio Video Bridging (AVB) Ethernet [17]. The AVB standards define two Stream Reservation (SR) classes called A and B for audio/video streams. Class A streams have higher priority than Class B streams. The Class A streams have maximum latency of 2 ms with a transmission period of 125 μs, while Class B streams have maximum latency of 50 ms with a transmission period of 250 μs over a maximum of 7 hops. Moreover, AVB defines six best-effort classes with priority scheduling among them. The AVB applies a Credit Based Shaper (CBS) using leaky buckets to control the traffic rate and the burstiness of streams while satisfying the QoS requirements. The best-effort traffic is sent without CBS.

AVB was a big step for providing QoS on Ethernet architecture. While it was enough for satisfying the QoS requirements of most audio/video traffic, it was not enough for the critical data flows that have stricter QoS requirements than audio/video streams. Therefore, the scope of the AVB task group was extended to cover time-sensitive transmission of data over deterministic Ethernet networks and it was renamed to Time-Sensitive Networking Task Group in 2012. TSN included the CBS shaper of AVB and extended them by adding Time-Aware Shaper (TAS), which is a time-division multiple access (TDMA) shaper to isolate the flows based on flow priority. TSN introduced a Control Data Traffic (CDT) class for the control traffic with high QoS requirements. TAS guarantees timely transmission of CDT [18]. TSN included the AVB and BE traffic classes defined in AVB Ethernet. The flows are classified into 802.1Q priorities. Each priority level is assigned to one or more time slots by TAS using a predefined scheduling table called Gate Control List (GCL). The slot assignments are stored in a TAS scheduling table that shows which queues can transmit in a time slot. As TSN can transfer the high priority CDT packets in dedicated



	CDT	AVB A	AVB B	Best Effort
Slot 0	1	0	0	0
Slot 1	0	1	1	1
Slot 2	0	0	0	0

Fig. 2. The TSN and AVB Ethernet scheduling.

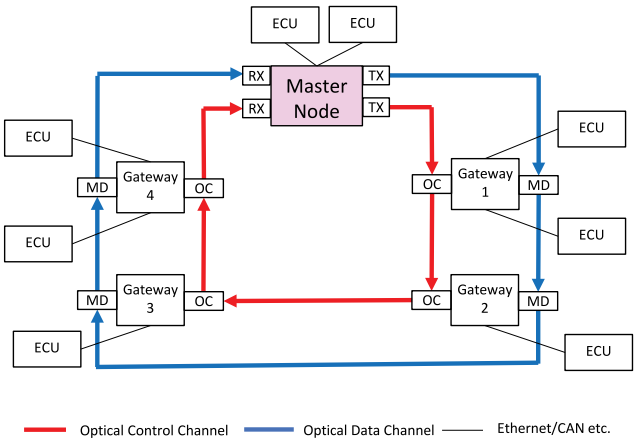


Fig. 3. The optical intra-vehicle backbone architecture.

time slots, the CDTs are completely protected from any jitter or delay that may be caused by AVB and BE traffic.

As an example, a basic time slot configuration of TSN is shown in Fig. 2.

1. In the first time slot (CDT slot), only CDT gates are open, so only CDT traffic can be sent to the link.
2. In the second time slot (AVB slot), all gates except the CDT gates are open, so all queues except the CDT queues can send packets to the link. AVB queues, which are controlled by CBS, can send traffic if they have enough credit. BE queue can send traffic when the link is idle.
3. In the third slot (guard-band slot), all gates are closed to prevent contention of control and other packets in the CDT slot.

There are only three time slots in the example above, but it is possible to set many time slots with different gate configurations in the same cycle. There may be multiple CDT slots in a cycle. Moreover, each link may have a different slot setting. The slot settings have a big impact on the performance of the TSN network. If the CDT slot is too short, control packets may get blocked and delayed by TAS. If the AVB slot is too short, the channel capacity may not be enough to carry data flows, which may cause congestion and packet drops. Calculating the optimum number of CDT and AVB slots and their lengths is an NP-hard problem, so there are works in the literature on their optimization by using Integer Linear Programming (ILP) and heuristics [19–23]. Configuring the GCLs in on each link may be difficult on a large scale network, so Cyclic Queuing and Forwarding (CQF) shaper was added to TSN standards to simplify the configuration [24]. Moreover, network-wide time synchronization may be difficult in some cases, so another shaper called Asynchronous Traffic Shaper (ATS) was added to TSN standards [25]. There are also proposals in the literature that use Software Defined Networking (SDN) for fast online TSN traffic scheduling using a global view of the network. For example, [26] proposes scheduling and routing time-triggered (periodic) flows with a pre-assumption of admission using around 50 slots in a TSN cycle. [27] jointly optimizes admission control, routing, and scheduling using around 70 slots in a cycle. While these TSN proposals in the literature show good results and can be applied to an optical packet switching architecture, it is difficult to apply them in SiPhON architecture, because a gateway cannot send data unless a slot is assigned to the gateway by the master node due to lack of LDs in the gateways and a gateway can send a slot request to the master node only by writing the request to another slot assigned to the gateway.

2.3. Optical backbone network

The fiber-optic links have many advantages over copper wires. The fiber-optic links are more resilient against environmental factors like electromagnetic waves and mechanical/chemical stress compared to copper wires. Moreover, fiber links do not radiate electromagnetic waves that may cause noise to other devices. Furthermore, fiber links weigh less while supporting higher-bandwidth than copper wires. Therefore, fiber-optic links have been used in intra-vehicle networks and avionics successfully for a long time [28]. For example, FlexRay supports both copper and fiber-optic links. MOST architecture uses only fiber-optic links. However, MOST supports maximum 150 Mbps and FlexRay supports maximum 10 Mbps bandwidth. While Ethernet supports high bandwidth over fiber-optic links, the current Ethernet hardware standards are not suitable to operate reliably in harsh working environments of vehicles. To cope with the high bandwidth requirements of recent applications like self-driving systems and to satisfy reliability requirements in harsh environments, Multi-Gigabit Automotive Optical PHYs (OMEGA) Study Group has been established [6]. OMEGA has been working on standardizing high speed optical Ethernet networks in intra-vehicle networks. OMEGA aims to achieve 50 Gbps per lane bandwidth communication with at least 15 meters of fibers in automotive temperature range (−40 to 125 °C). To the best of our knowledge, there are no works on OMEGA published in the literature so far. An intra-vehicle network needs to be environmentally resistant against low and high temperatures, but the life of vertical surface emitting laser (VCSEL) diodes used in the transceiver module of OMEGA are dramatically shortened when operated at a temperature of more than +100 °C even with improvements such as a quantum dot active layer [7]. In OMEGA optical Ethernet backbone network, all backbone links of all gateways require LDs to generate optical packets. The failure probability increases with increasing number of LDs, which may bring a high cost and reliability problems in the long-term.

3. SiPhON intra-vehicle architecture

Satisfying the severe hardware reliability requirements of intra-vehicle networks while providing high-bandwidth and low latency by Ethernet may be costly. As a possible solution, we propose an optical cut-through intra-vehicle backbone architecture called Si-based In-vehicle Photonic Network (SiPhON), which may have a lower cost than an optical Ethernet backbone architecture in a vehicle. In the SiPhON architecture, there is a single master node that controls the gateway nodes and there are multiple gateway nodes that switch data between the ECUs and the backbone network. The backbone nodes in SiPhON are connected by unidirectional optical links that form a

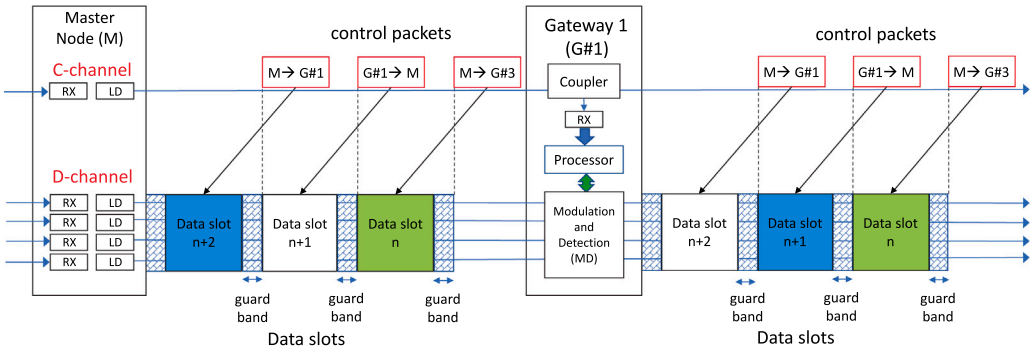


Fig. 4. An example of incoming and outgoing control packets and data slots in gateway 1.

ring topology as shown in Fig. 3. There is a single control channel for exchanging control packets and a data channel to carry data packets between the master node and gateway nodes. The control and data channels use cut-through switching to minimize the latency. It is a slot switching architecture where slots carry packets between the master node and gateway nodes in the data channel as shown in Fig. 4. The gateways in SiPhON do TDMA switching with a granularity of a slot, where each slot is used for transmission between the master node and a single gateway, so the meaning of slot in SiPhON is different from TSN Ethernet where a slot means a period of time dedicated to a given list of priority classes. The slots in SiPhON are assigned to the gateway nodes by the master node according to a slot scheduling algorithm. An assigned gateway can use the slot for sending or receiving data packets. When a slot is assigned to a gateway node, the master node informs the gateway node by sending a control packet before the data slot. The assigned gateway processes the control packet and configures its hardware to receive or send data using the upcoming slot in the data channel. As the processing and configuration in the assigned gateway takes some time, there is a guard-band time between the data-slots and also between the control packets and the associated data slots. When a gateway receives a control packet, it processes the information in the control packet and configures its hardware accordingly during this guard-band time for the upcoming data slot. Also, after the data slot ends, the gateway reconfigures its hardware during the guard-band time until the next slot arrives.

In SiPhON architecture, laser diodes (LD) are carried only in the master node. The master node generates the optical control packets and data slots for data packets by LDs and sends them to the gateways. Data channel LDs work in parallel to generate a slot. Each data channel LD generates a part of the slot. The gateway nodes do not have LDs, so they cannot generate optical signal to send data packets. The master node generates the light in the slots. The gateways inject their data packets into the slots by modulating the light in the slots by using optical modulation and detection (MD) circuits. The silicon photonics hardware architecture of the MD circuit, which is a Mach–Zehnder Interferometer (MZI) switch integrated with a photo diode (PD), was presented and demonstrated in [7]. To receive the control information on the control channel the gateway nodes use an optical coupler (OC) to sample and read the control packets. The gateway nodes do not generate control packets, so they do not need LDs on the control channel, either. Only the master node uses LDs for generating control packets. As fewer LDs used are in a SiPhON network, the LD failure probability and the long-term cost can be decreased compared to Ethernet-based architectures like OMEGA, which require LDs in each gateway to generate optical packets.

While using fewer LDs can decrease the cost and failure probability, the gateways cannot send packets to the optical backbone directly when they want because they lack LDs. The master node assigns slots to the gateways and informs the gateways by sending control packets right before the data slots in the control channel. The types of actions defined in control packets are as follows:

- Listen: The gateway node selected by the master node will receive data from the master node in the associated data slot.
- Talk: The gateway node selected by the master node may send data to the master node in the associated data slot.
- Idle: The associated data slot is not used.

To illustrate how SiPhON works, the processing of the control packet and data slots in gateway 1 is shown in Fig. 4 as an example. After the data slots and control packets are generated in the master node, they are sent to the first gateway node called gateway 1 in the ring topology. The left side of gateway 1 shows the packets and slots before entering the gateway, while the right side of gateway 1 shows the same packets and slots after leaving gateway 1. The slots carrying the data packets to/from gateway 1 are shown in blue, while the slots carrying data packets to/from other gateways are shown in green. The slots without a data packet are shown in white. The first control packet entering gateway 1 contains the information M→G#3, which means that this is a Listen packet for gateway 3. This information shows that the upcoming associated data slot, which will arrive after a guard-band time difference, is for another gateway, so gateway 1 does not read the data in the data slot and the data slot leaves gateway 1 as it is. The second control packet contains the information G#1→M, which means that this is a Talk control packet for gateway 1. Gateway 1 can use the upcoming associated data slot to forward a data packet from the connected ECUs to the master node. In case there is a packet in its buffer to send, gateway 1 reconfigures its switching fabric and encapsulates the packet in a frame, and injects into the upcoming associated data slot by using the MD circuit. Gateway 1 can also write feedback information like its buffer stats to the frame header in the slot so that the master node can learn the current status of the gateway. Master node may send Talk control packets to gateways to periodically sample the current status of the gateways to give a better decision when scheduling the slots. The third control packet contains the information M→G#1, which means that this is a Listen packet for gateway 1. There will be a data packet to this gateway from the master node in the upcoming data slot, so the gateway node updates its switching fabric during the guard-band time. Then, it forwards the upcoming data slot to its MD circuit and extracts the frame in the slot by converting the optical signal to the electronic domain by the MD circuit. It decapsulates and forwards the data packet to the destination ECU.

If a source node connected to a gateway wants to send a packet to a destination node connected to another gateway, first the gateway sends the packet to the master node in a Talk slot. Then the master node sends the packet to the destination gateway in a Listen slot. Finally, the gateway forwards the packet to the destination node.

The gateways and master node should be synchronized for the slotted architecture. The Precision Time Protocol (PTP) may be used for the time synchronization among the nodes [29]. Moreover, the control packets and data channel slots should be synchronized. The

synchronization in SiPhON is under development, so its hardware and processing requirements are currently unknown.

In this paper, we used a fixed slot size and assumed that a single packet is carried in a slot for simplicity, but it may cause bandwidth inefficiency when there are many small packets. Possible solutions like carrying multiple packets in a slot will be explored in future works. Moreover, using a single master node or a single path between backbone nodes may cause a single point of failure, but hardware failure scenarios are out of the scope of this paper. Increasing the resilience of the architecture against failures by adding backup hardware and links is left as a future work.

4. Slot scheduling in SiPhON

To decrease the probability of LD failures in the backbone, the gateways in SiPhON architecture do not carry LDs. Therefore, the gateways cannot generate light to send packets to the backbone directly, which limits the communication from the gateways to the master node. To send a packet from the gateway to the master node, the master node should generate an optical signal by its LDs and send it to the gateway in a slot so that the gateway can inject its packet into the slot by modulating the optical signal in the slot. Moreover, when a gateway wants to send packets to the backbone, the only way that the gateway can inform the master node and ask for slots is to write the request to the header of a previous Talk slot assigned to the gateway. If there is no Talk slot scheduled to the gateway, the gateway cannot inform the master node that it has a packet to send. The only way for a master node to check if there is a new packet in the gateway is to poll the gateway node by sending an empty Talk slot to the gateway. However, if there is no packet to be sent to the master node in the gateway, the slot used for polling the gateway becomes unused, which decreases bandwidth efficiency.

In our initial work, to show the basic operation of the architecture, we used a manually assigned fixed and periodic pattern of slot schedule list based on the average amount of traffic sent to the backbone by the gateways [12]. The advantage of fixed and periodic slot schedule list is that the master node does not need feedback from the gateways like slot requests or polling. It works as long as the gateways and the master node do not use bandwidth more than the bandwidth assigned by the slot schedule. For example, the periodic and fixed slot schedule {Listen, Talk GW1, Talk GW2, Talk GW3, Talk GW4, Talk GW5, Talk GW3} in [12] assigns around 28.6 Gbps to GW3 and 14.3 Gbps to the rest of the gateways and the master node out of 100 Gbps bandwidth capacity. However, using a fixed and periodic slot schedule list has the following shortcomings:

- A periodic and fixed slot schedule assignment may cause unnecessarily high delays when the traffic is bursty. As an example, assume that the periodic and fixed slot schedule {Listen, Talk GW1, Talk GW2, Talk GW3, Talk GW4, Talk GW5, Talk GW3} in [12] is used. In this case, the master node and the gateways except gateway 3 can send a packet only once in a period of 7 slots. Even if all the ECUs connected to a gateway or master node generate completely smooth and periodic traffic, the overall traffic to be sent to the backbone by the gateway or master node may not be smooth and periodic. Packet contentions may occur at the gateway, causing a bursty traffic. In case the master node or a gateway except gateway 3 has three packets in its output buffer to the backbone, the first packet can be carried by the first slot assigned to the gateway in the first period. Then, the second packet in the queue would be carried by the slot assigned to this gateway in the next period after waiting 7 slots time. Moreover, the third packet in the queue would be carried by the slot assigned to this gateway in the period after the next period after waiting 14 slots time. The delays would increase at a fast pace with the increasing burstiness of the traffic.
- High priority packets in a gateway may end up waiting for the transfer of the low priority packets or idle slots of other gateways until the gateway's turn comes in the slot schedule list.
- The fixed and periodic slot scheduling is not adaptive to the changes in the traffic matrix. Some ECUs may generate variable traffic like Infotainment, Internet, and application updates, which may change the traffic matrix and cause bandwidth inefficiency and higher packet delays.
- A fixed and periodic slot schedule does not take the traffic patterns of flows like packet generation periodicity into account.

In this paper, we extend [12] with a dynamic slot scheduling algorithm that can decrease the end-to-end packet latency of flows in SiPhON architecture by solving the shortcomings of fixed and periodic slot scheduling. We called the algorithm dynamic because, unlike fixed and periodic slot scheduling, it dynamically assigns the slots to the backbone nodes based on the reservation requests and buffer occupancy feedbacks without requiring traffic matrix information. The key ideas of the new dynamic slot scheduling algorithm are as follows:

- Decreasing the slot waiting times of packets of high priority flows that generate packets with a known (fixed or variable) interval by calculating their arrival times to gateways.
- Decreasing the slot waiting times of bursty low priority flows by sampling the buffer occupancy of gateways.
- Adaptively changing the slot assignment ratio and thus the bandwidth ratio of the nodes based on the reservation requests and the buffer occupancy of gateways.

Many traffic sources like sensors in a vehicle generate packets with a known interval. For example, an engine sensor checks the status of the engine periodically and sends time-sensitive sensor data in fixed periodic control packets. As the control packets are generated periodically, the traffic source can calculate the departure time of the future packets. Also, there is growing interest in variable (adaptive) rate sampling in the literature. Changing the sampling and packet generation intervals based on the state of the vehicle can decrease storage costs when the generated data is logged in a storage, decrease transmission costs when the generated data is shared with the environment over a mobile connection, and ease processing and energy consumption [30]. For example, [31] proposes changing the GPS sampling rate based on the movement (free flow, acceleration, stop, etc.) of the vehicle to decrease the transmission and storage costs. Another example is given by [32] where the sampling rate of sensors changes with the aggressiveness of drivers (intensity of breaking and acceleration) and the current speed. Even when the sampling and packet generation rate is variable, a traffic source can calculate the departure time of the next packet if the next sampling time is decided when the last sample was taken. For example, if a GPS sensor is changing the sampling rate based on the vehicle speed in the last sample, the next sampling and the packet generation time become clear in the last sampling. Our dynamic slot scheduling algorithm decreases the slot waiting times of packets of high priority time-sensitive flows that generate packets with a known (fixed or variable) interval by reserving slots to the upcoming packets by calculating the optimum time to send slots based on the interval information sent by flow sources to the master node.

Low priority flows like best effort Internet traffic may be bursty and have variable traffic rates, which may cause buffer buildups in the gateways and high delays. To decrease the slot waiting times of bursty low priority flows, the dynamic slot scheduling algorithm samples the buffer occupancies in gateways. It assigns the unreserved idle slots to the gateways with high buffer occupancy. Therefore, the bursty traffic arrivals get a high portion of the idle slots, which allows to serve them faster. Moreover, idle slot assignment based on the buffer occupancy effectively provides a dynamic bandwidth allocation to the gateways without requiring traffic matrix information, unlike the fixed and periodic slot assignment.

When a traffic source node (ECU or sensor) sends a packet of a flow that generates packets with a known (fixed or variable) interval, it adds a SiPhON packet header to the packet. It writes the flow identifier and the next interval time, which is the time difference between the departure time of the current packet and the expected departure time of the next packet. If the interval is fixed, it may be enough to write it only to the first packet of the flow and save the interval information in the master node. If the source is connected to a gateway, the gateway reads this header and stores it until the packet is sent to the backbone. When a packet in the gateway buffer is injected into a slot, the gateway first writes its buffer occupancy information to the slot header. If the injected packet has a SiPhON packet header, the gateway also copies the SiPhON packet header to the slot header and writes the time difference between the arrival and departure time of the packet to the slot header. When the packet arrives at the master node, the master node reads the slot header. If there is an interval time information, the master node calculates the optimum time to send a slot to carry the next packet of this flow in the backbone so that the slot can arrive at that gateway right after the next packet of this flow arrives at the gateway. After checking the previous reservations in the reservation list and the flow priority levels of the packets, the master node reserves the first slot that can send the packet after the calculated time.

In case the flow source node is connected to a gateway, and it wants to send a packet of a flow with a known interval time to the master node, the slot reservation for the next packet of the flow by the master node is done as follows:

1. The flow source writes its flow id and the latest interval time denoted by D_{next}, which is the time difference between the departure time of the current packet and the expected departure time of the next packet, to the SiPhON header of the current packet.
2. When the packet arrives at the gateway, the gateway stores the arrival time of the packet.
3. When a data slot for carrying this packet on the backbone arrives at the gateway, the gateway writes the flow id and the time difference between the arrival and departure time of the packet, which is denoted by G_{wait}, to the slot header. This time difference shows how long the packet waited in the gateway due to buffering, processing, etc. Moreover, the gateway writes the buffer occupancy of its priority queues to the slot header.
4. The packet is encapsulated in a frame and sent to the master node over the backbone by injecting it into a data slot by the MD circuit in the gateway.
5. The data slot circulates the backbone ring network and returns to the master node.
6. The master node decapsulates the frame in the slot and reads D_{next} and G_{wait} information in the headers.
7. The master node keeps a record of the departure times of the slots. Let us denote the departure time of this returning data slot that carried the packet to the master node as S_{prev}. The master node calculates the optimum time to send a data slot for the next packet of this flow, which is denoted by S_{next}, by the formula

$$S_{next} = S_{prev} + D_{next} - G_{wait}. \tag{1}$$

The departure time difference between a control packet in the control channel and the associated data slot in the data channel in SiPhON is fixed. It equals the sum of the guard-band duration, which is denoted by d_{guard}, and the control packet duration, which is denoted by $d_{control}$. The optimum departure time of the control packet in the control channel for this data slot, which is denoted by C_{next} can be calculated by.

$$C_{next} = S_{prev} + D_{next} - G_{wait} - d_{guard} - d_{control}. \tag{2}$$

8. After checking the previous reservations in the reservation list and the flow priority levels of the packets, the master node reserves the first suitable slot that it can send for the packet after the calculated time.

When the master node sends a reserved slot to a gateway, it writes the flow id of the flow source, which did the reservation, to the control packet of the slot. The gateway injects the packet of that flow into the slot by MD circuit.

When a flow source node is connected to the master node, and it wants to send a packet to a destination node connected to a gateway node, first, the source node sends the packet to the master node. The master node checks if there is a slot reservation for this packet. If there is a reservation, the master node delays the packet until the reserved slot is generated. Then the master node sends the packet in the reserved slot to the gateway. If there is no reservation for this packet, after checking the reservations in the reservation list and the flow priority levels of the packets, the master node reserves the earliest slot that can send the packet.

In case the flow source node is connected to the master node and it wants to send a packet of a flow with a known interval time to a gateway, the slot reservation for the next packet of the flow by the master node is done as follows:

1. The flow source writes its flow id and the time difference between the departure time of the current packet and the expected departure time of the next packet, which is denoted by D_{next}, to the SiPhON packet header and sends the packet to the master node.
2. When the packet arrives at the master node, the master node reads the SiPhON packet header.
3. The master node calculates when the next packet of this flow will arrive at the master node, which is denoted by P_{next}, by adding D_{next} to the arrival time of the last packet of the flow, which is denoted by P_{prev}, by the formula

$$P_{next} = P_{prev} + D_{next}. \tag{3}$$

The optimum departure time of the control packet in the control channel for this data slot can be calculated by.

$$C_{next} = P_{prev} + D_{next} - d_{guard} - d_{control}. \tag{4}$$

4. After checking the previous reservations in the reservation list and the flow priority levels of the packets, the master node reserves the first slot that it can send for the packet after the calculated time.

The packets arrive at the gateways continuously without synchronization, but the data slots in the gateway have a fixed length and they are sent in a TDMA fashion, so it is usually not possible to send a control packet for this gateway exactly at the calculated optimum time C_{next}. Therefore, the master node reserves the first suitable slot after C_{next}. Moreover, G_{wait} may be higher than D_{next}, which may happen when the packet containing the latest feedback was delayed in the gateway due to queuing, processing, etc. more than interval time D_{next}. In that case, the master node reserves the first suitable slot in the reservation list by checking the reservations and the flow priorities.

In this paper, when a new packet arrives with a reservation request for the next packet of the flow, the optimum slot for the new reservation and also the next slots may have been reserved for other packets. In that case, the reservation for the new packet is scheduled at the first slot that is idle or that has a reservation to a packet of a lower priority flow. The contending reservations for the flows with a lower priority level are delayed in the slot reservation list. Fig. 5 presents an example slot reservation by the master node when the master node receives a new packet from a source connected to the master node or receives a returning slot that has interval time information. The slots show the

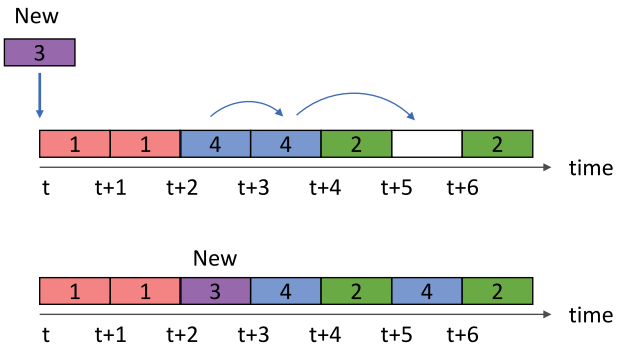


Fig. 5. An example of slot reservation.

reservations after time t. The numbers inside the slots show the priority level of the flows that reserved the slots where lower values denote lower priority. There is an empty reservation at time $t+5$. Assume that the master node wants to reserve a slot for a packet of flow with priority level 3 after time t in the reservation list. As the slots at t and $t+1$ are assigned to flows with higher priority, the reservation is done to slot at $t+2$. The reservations at $t+2$ and $t+3$ are delayed accordingly without affecting the reservations of higher priority flows.

When the master node sends a new slot, it assigns the slot to a gateway or the master node by a slot assignment algorithm. The pseudocode code of the dynamic slot assignment algorithm that is used in this paper is shown in Fig. 6. The algorithm can be modified based on the prioritization requirements of the network. In the pseudocode, n denotes the number of gateways in the backbone. The array $last\_assigned\_time[k]$ stores the last time a Talk slot was assigned to each node. Using the local buffer information in the master node and the latest buffer statistics sent by the gateway nodes in the slot headers, the master node finds the node that has the highest number of packets in the highest priority non-empty queue and stores in $highest\_queue\_node$. By the loop on line 4, the master node first checks the last time a slot was assigned to each node and finds the gateway that has the oldest reservation time, and saves its id in $oldest\_node$. The master node also saves the last time that a Talk slot was assigned to this node in $oldest\_time$. If no slots have been assigned to this gateway for a duration longer than a threshold, a Talk slot is assigned to this gateway to check if this gateway has a packet to send and to receive the latest buffer occupancy information of this gateway. If there is a reservation to another node in the reservation list, it is delayed to the next slot. If none of the gateways has passed the threshold, the algorithm checks if there is a reservation to a node in the reservation list. If there is a reservation, the slot is assigned to that node. If not, the algorithm assigns the slot to the node that has the highest number of packets in the highest priority level queue. If none of the nodes has a packet in their buffers, the algorithm assigns the slot to the gateway that the longest time has elapsed since its last slot assignment, which was found by the loop on line 4.

In this paper, we used strict priority queuing and scheduling based on flow priorities. Traffic shapers like TAS and CBS in TSN Ethernet can provide better prioritization and harness the traffic burstiness inside a network. The scope of this paper is minimizing the slot scheduling delays in SiPhON, so the design of traffic shapers for SiPhON is left as a future work.

The dynamic slot scheduling requires the expected departure time of the next packet, flow id, the buffer waiting time, and the buffer occupancy information in the gateway as feedback. This extra information is sent in the slot headers by the gateways. The specifications are not decided yet, but we can expect this header size to be around 10–30Bytes. In the case of a slot size of 1600 Bytes, the bandwidth required for headers of dynamic slot scheduling may use around 1%–2%, which may be negligible.

Sending the expected departure time of the next packets from the application layer to the hardware layer in a source node may increase the cost and the processing delays. Their impact will be evaluated when a prototype of the SiPhON is implemented. If the cost or processing delays due to dynamic slot assignment are too high, the network can use fixed and periodic slot scheduling, which does not require the expected departure time of the next packets and buffer occupancy information of the nodes.

5. Simulation

We simulated the intra-vehicle network shown in Fig. 7 with a traffic matrix inspired from [33]. In the simulation scenario, the traffic matrix had 12 flows (7 video flows, 1 best-effort flow, and 4 control flows). The packets of video flows were paced at the source. The traffic sources were as follows:

1. Six video cameras (CAM) send uncompressed 4K30p smooth video traffic (6 Gbps) to Dashcam. These video flows send 1448 Bytes data every around 2 μs.
2. The Dashcam sends uncompressed 4K30p smooth video traffic (6 Gbps) to Head Unit. This video flow sends 1448 Bytes data every around 2 μs.
3. The Infotainment unit sends 6 Gbps best-effort traffic to the Head Unit. This data flow sends 1448 Bytes data with an exponential distribution of 2 μs.
4. The Control unit sends control packets to the Dashcam. This control flow sends 46 Bytes control messages with a variable interval of 30 μs during an emergency and 120 μs otherwise.
5. The Dashcam unit sends warnings to the Control Unit. This control flow sends 46 Bytes control messages every 0.5 ms.
6. The Head Unit sends 46 Bytes control messages to the Control Unit every 0.5 ms.
7. The Control Unit sends 46 Bytes control messages to the monitor of Head Unit every 0.5 ms.

There are three types of traffic sources in the simulations. They are

1. The control unit sends control packets to the Dashcam with a variable but known interval. Normally the interval was 120 μs, but it was decreased to 30 μs in between 1 ms and 2 ms in simulation time. For the dynamic slot scheduling algorithm, the flow source sends the interval time of the next packet to the master node each time it sends a packet.
2. The Infotainment unit sends best-effort bursty traffic to the Head Unit without any interval information.
3. The rest of the flow sources send packets with a fixed and known interval. For the dynamic slot scheduling algorithm, the flow sources send the interval time of the next packet to the master node each time they send a packet.

We simulated the TSN Ethernet architecture by using computer-based CoRE4INET simulator [34]. The parameters of TSN Ethernet simulation were as follows. The nodes were connected by a ring topology with 100 Gbps bidirectional optical links. The shortest path routing was applied. CoRE4INET simulator [34] does not support cut-through switching, so store-and-forward switching was used. The gateway node denoted by master in Fig. 7 was an ordinary TSN Ethernet gateway like the other gateways. The MTU was 1500 Bytes. The total UDP and Ethernet header size was 52 Bytes. Even when cut-through switching is used, Ethernet switches impose port-to-port forwarding delays to the packets while processing and routing the packets inside the switch. CoRE4INET simulator [34] has a default value of 8 μs per hop for the forwarding latency of TSN Ethernet switches. In [12], we used the default value of CoRE4INET simulator in our simulations. However, recent Ethernet switches with fast link speeds have lower processing times. We saw that there are some 100 Gbps Ethernet switches with sub 0.2 μs latency

```
n ← number of gateways
last_assigned_time[k] ← last time a Talk slot was assigned to node k
highest_queue_node ← the node that has highest number of packets in
the highest priority level queue
for k ← 1 to n do
    if last_assigned_time[k] < oldest_time then
        oldest_node ← k
        oldest_time ← last_assigned_time[k]
    end if
end for
assign_node ← −1 (assign the slot to this node)
if NOW − last_assigned_time[oldest_node] < threshold then
    assign_node ← oldest_node
    Delay_Reservation()
else if reserved_slot_node ≠ NULL then
    assign_node ← reserved_slot_node
else if highest_queue_node ≠ NULL then
    assign_node ← highest_queue_node
else
    assign_node ← oldest_node
end if
```

Fig. 6. The pseudocode of dynamic slot assignment algorithm.

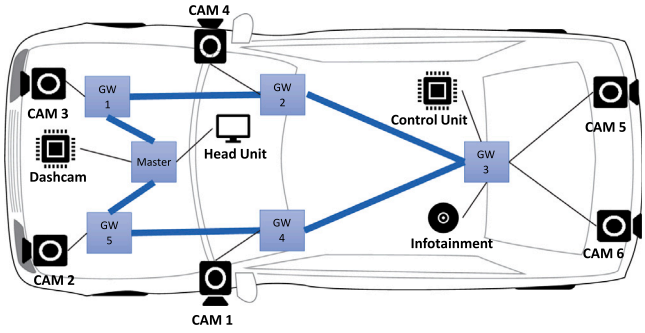


Fig. 7. Simulated intra-vehicle network topology.

in the market, but they are not TSN capable and they are extremely expensive. We saw that some of the low-latency TSN switches state that they have sub-microsecond latency without giving an exact number. The lowest latency TSN switch with a delay specification that we could find was from Cast Inc., which has 0.92 μs latency [35]. Therefore, we set the forwarding latency of TSN switches to 0.92 μs in the simulations. There were three slots in one TSN cycle in all links using the same TSN configuration as explained in 2.2. In [12], the length of the CDT time slot in the gateways was set to 35 μs because the maximum packet transmission, propagation and Ethernet forwarding latency was around 33 μs in the network. In this paper, the length of the CDT time slot in the gateways was set to 5 μs because the maximum packet transmission, propagation, and Ethernet forwarding latency was decreased to around 4.1 μs due to lower Ethernet forwarding latency. The duration of the guard band after CDT was set to around 0.12 μs, which equals the transmission time of an Ethernet frame with 1500Bytes payload over 100 Gbps links. At every 30 μs, a CDT slot was scheduled. The time between CDT slots was assigned to AVB and best-effort flows. The video flows were assigned to AVB A priority. AVB traffic uses at most 75% of the link bandwidth.

We simulated our SiPhON architecture by a computer-based simulator that we implemented on OMNeT++ framework. The simulation parameters of our architecture were as follows. The optical backbone links carried a 100 Gbps data channel and a 1.25 Gbps control channel. The links in the backbone were unidirectional and the transfer in the ring topology was in the clockwise direction. As the TSN simulations used bidirectional links with shortest path routing, TSN simulations had advantages in terms of bandwidth and link utilization over SiPhON simulations. In TSN simulations, the traffic from GW4, GW5, and the master node routed the packets in THE clockwise direction, while the rest of the gateways routed in the counterclockwise direction. Therefore, in TSN Ethernet simulations, the highest link utilization was around 30%, which was on the 100 Gbps link from GW1 to master with around 30 Gbps traffic rate. On the other hand, in the SiPhON architecture, the highest link utilization was around 48%, which was on the 100 Gbps link from GW5 to master with around 48 Gbps traffic rate. The backbone network applied cut-through switching and the edge links applied store-and-forward switching. The gateway node denoted by master in Fig. 7 was the master node that assigns the slots. In [12], the guard band between the slots was 100 ns, which was a bit conservative. In this paper, we set the guard band to 50 ns. The maximum transmission unit (MTU) of a slot was 1500 Bytes. The maximum video packet size was set to fully use the 1500 Bytes MTU.

We simulated a SiPhON network with the dynamic slot scheduling algorithm and the fixed and periodic slot scheduling. We compared the end-to-end packet delays of different types of flows. We also simulated a TSN Ethernet network only as a reference to show the total delays including traffic shaping and processing algorithms delays in a commercially available architecture. Direct comparison of the delay results in TSN and SiPhON is not fair, because TSN Ethernet used TAS and CBS traffic shapers that further delayed packets in the simulation and it has processing delays. For a fair comparison, SiPhON should be simulated with traffic shaping and processing delays. However, the design of traffic shapers for SiPhON is left as a future work, and the processing delays in the SiPhON architecture cannot be known until a prototype of the device is built. We simulated the TSN architecture only to show that the extra delays due to the slot scheduling in our architecture may be negligibly low when compared with the total delays in TSN Ethernet, which includes the delays due to traffic shaping and processing algorithms.

In the static slot scheduling simulations with SiPhON, the slot scheduling algorithm applied a manually assigned fixed set of 7 slots {Listen, Talk GW1, Talk GW2, Talk GW3, Talk GW4, Talk GW5, Talk GW3} that repeats in cycles. The first Listen slot was used by the master node to send data to the gateways. The other Talk slots were used for transferring data from the indicated gateways to the master node. As there were two video cameras connected to gateway 3, two slots

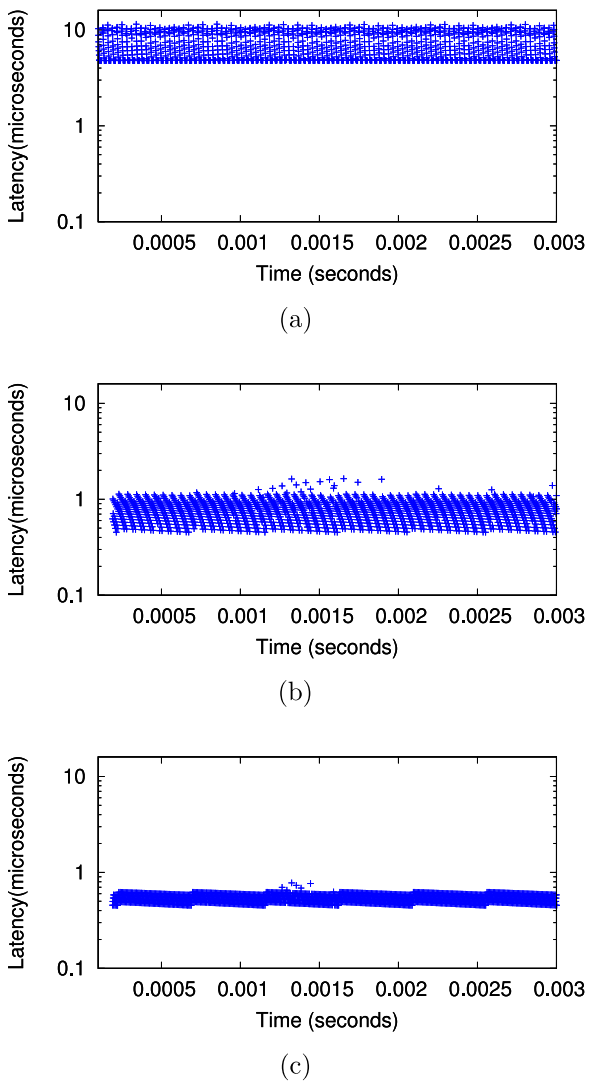


Fig. 8. The comparison of latencies of video packets from CAM 5 to Dashcam in (a) TSN Ethernet, (b) SiPhON with fixed and periodic slot scheduling, and (c) SiPhON with dynamic slot scheduling.

were assigned to transfer data from this gateway in a slot cycle, which effectively made the assigned bandwidth to traffic from this gateway twice the assigned bandwidth to traffic from other gateways. In the dynamic slot scheduling simulations, the slot scheduling was based on the dynamic slot scheduling algorithm introduced in this paper.

First, we evaluated the end-to-end packet latency of a video flow. shows the packet latencies of video traffic from CAM 5 to Dashcam. The x-axis is the simulation time in terms of seconds and the y-axis is the packet latencies in terms of microseconds in all simulation results. The y-axis is plotted in a logarithmic scale. Fig. 8(a) shows the total latencies due to transmission, processing, and shaping in TSN Ethernet. There is no slot scheduling in TSN Ethernet, so there is no slot scheduling delay. The simulation results show that the packet latencies were between 5 μs and 12 μs. The minimum latency of around 5 μs was mainly due to the forwarding latency of Ethernet switches and the rate control of CBS. The maximum latency of around 12 μs was mainly due to the buildup of AVB slot queues when the video traffic was blocked by TAS while control packets were transferred in CDT slot by TSN Ethernet [36]. The TSN architecture was simulated with store-and-forward switching because CoRE4INET simulator [34] simulator does not support cut-through switching. However, it is possible to estimate the delay of cut-through switching by subtracting the extra transmission delays. When store-and-forward switching is used, transmission delays occur at five links. If only cut-through switching was used along the path between the edge nodes, the transmission delays would occur at only one link, so the latencies would be around $4 * (1500 * 8/10^{11}) = 0.48\mu s$ lower. Considering the amount of current end-to-end latency, using cut-through switching would not make a big difference. Figs. 8(b) and 8(c) show the video packet latencies when fixed and periodic scheduling and dynamic scheduling algorithms were used, respectively. When the fixed and periodic slot scheduling was used, most of the latencies were between 0.4 μs and 1.6 μs. On the other hand, when the dynamic slot scheduling algorithm was used, most of the latencies were between 0.4 μs and 0.8 μs, which was lower than the fixed and periodic slot scheduling. The delays increased a bit during the time (1 ms,2 ms) because of the increased traffic from the control unit to the Dashcam. The packet delays due to slot scheduling in SiPhON were much lower than the total packet delays in TSN Ethernet, which includes traffic shaping and processing delays, so we can say that the extra slot scheduling delays due to limitations in the SiPhON architecture can be made negligibly low by using the dynamic slot scheduling algorithm. As a prototype of the SiPhON device has not been built yet, the processing delays in SiPhON are unknown and there is no traffic shaping, so the overall end-to-end packet latency in SiPhON architecture cannot be evaluated, yet. The overall end-to-end packet latency in SiPhON architecture will be evaluated after the prototype of the SiPhON device is built.

Next, we evaluated and compared the end-to-end packet latency of a control flow. Fig. 8(c) shows the packet latencies of video traffic from Control Unit to Dashcam. The control unit sent control packets to the Dashcam at a variable but known interval. Initially, the packet interval was 120 μs, but it was decreased to 30 μs in between 1 ms and 2 ms, then again increased to 120 μs. Fig. 9(a) shows the latencies in TSN Ethernet. As TSN carries the control packets in a dedicated slot, all control packets had a fixed latency of 4.2 μs, which is mainly due to the processing delays of TSN switches. Fig. 9(b) and Fig. 9(c) show the video packet latencies when the fixed and periodic slot and dynamic scheduling algorithms were used, respectively. When the fixed and periodic slot scheduling was used, most of the latencies were between 0.25 μs and 0.89 μs. On the other hand, when the dynamic slot scheduling algorithm was used, most of the latencies were between 0.23 μs and 0.39 μs. Again, the dynamic slot scheduling algorithm gave lower latencies than the fixed and periodic slot scheduling.

Finally, we evaluated and compared the end-to-end packet latency of a best-effort flow. Fig. 9(c) shows the packet latencies of best effort traffic from Infotainment ECU to Head Unit. Fig. 10(a) shows the latencies in TSN Ethernet. Most of the latencies were between 4.8 μ and 11 μs. The blocking and buffering of video packets during CDT slots increased the delays. Fig. 10(b) and Fig. 10(c) show the video packet latencies when fixed and periodic slot scheduling and dynamic scheduling algorithms were used, respectively. When the fixed and periodic slot scheduling was used, most of the latencies were between 0.4 μs and 26 μs. On the other hand, when the dynamic slot scheduling algorithm was used, most of the latencies were between 0.46 μs and 2.4 μs. Unlike in the fixed and periodic slot scheduling, which assigns a fixed ratio of the bandwidth to a gateway, the best-effort flows are not bounded with a portion of the bandwidth in dynamic slot scheduling, so the best effort traffic could get more bandwidth and get lower packet delays.

6. Conclusion

In this paper, we presented a novel optical intra-vehicle backbone network architecture that may have a lower cost and higher reliability in terms of hardware. However, our architecture may cause higher packet delays than Ethernet due to the lack of LDs in the gateways. The overall delays in SiPhON with the slot scheduling, packet processing, and traffic shaping delays may be higher than in Ethernet,

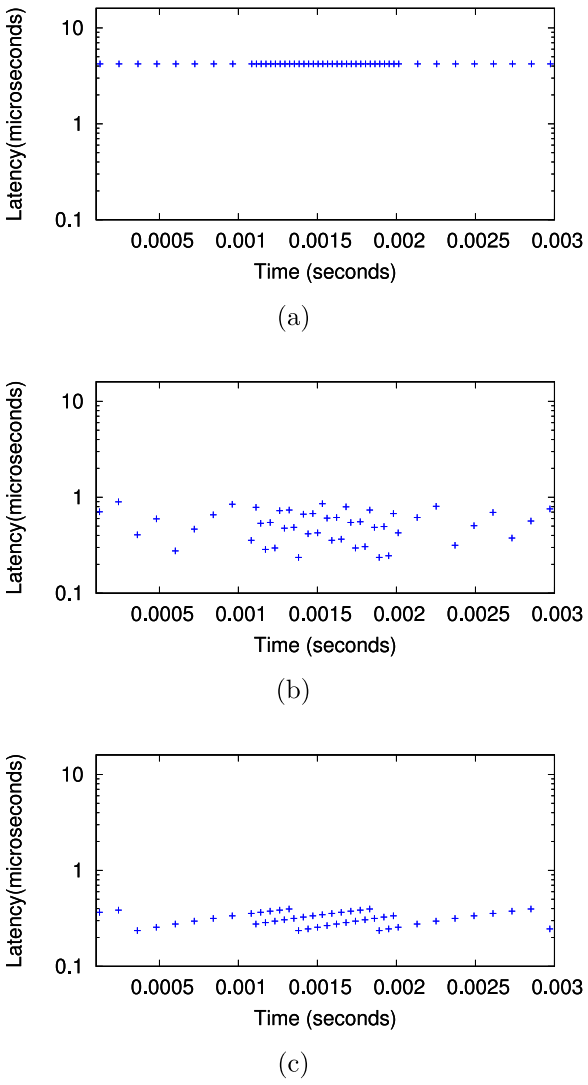


Fig. 9. The comparison of latencies of control packets from Control Unit to Dashcam in (a) TSN Ethernet, (b) SiPhON with fixed and periodic slot scheduling and (c) SiPhON with dynamic slot scheduling.

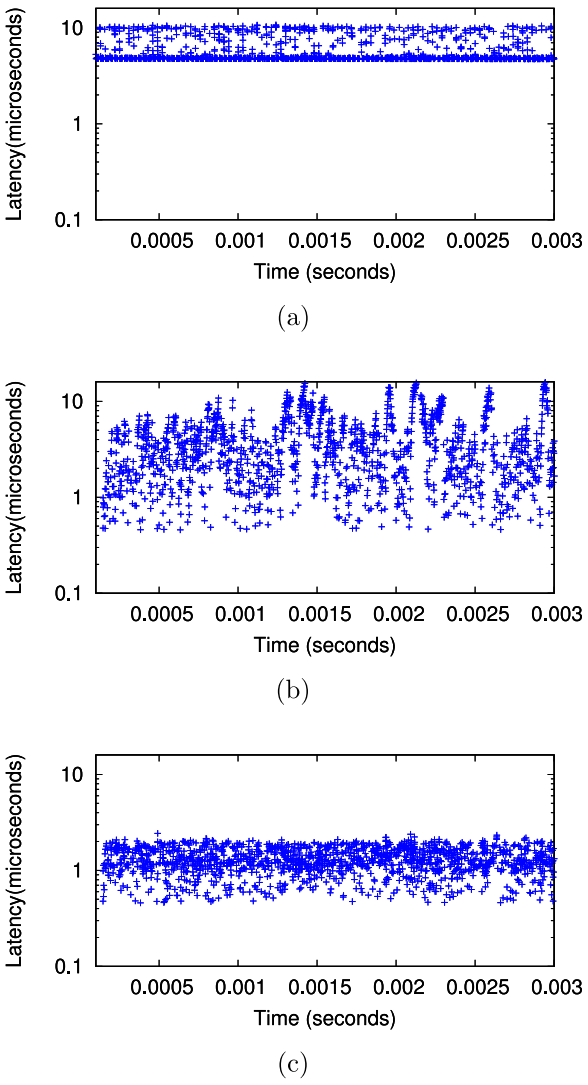


Fig. 10. The comparison of latencies of best effort packets from Infotainment ECU to Head Unit in (a) TSN Ethernet, (b) SiPhON with fixed and periodic slot scheduling and (c) SiPhON with dynamic slot scheduling.

but SiPhON may have a better performance/cost ratio than Ethernet if it can achieve a lower cost and higher reliability. In this paper, we proposed two slot scheduling methods and showed that a dynamic slot scheduling algorithm that can achieve lower packet latencies than fixed slot scheduling. Moreover, we showed that the packet delays due to dynamic slot scheduling in SiPhON are negligibly low when compared to the total delays including traffic shaping and processing delays in a TSN Ethernet intra-vehicle architecture. As a SiPhON prototype has not been implemented yet, the processing delays in SiPhON are currently unknown. Moreover, there are many traffic shaping algorithms in the literature so evaluating their delays in our architecture requires separate work. As a future work, we will calculate the processing delays in SiPhON after a prototype is built. We will also evaluate the delays of traffic shaping algorithms in SiPhON architecture.

CRediT authorship contribution statement

Onur Alparslan: Conceptualization, Methodology, Software, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Visualization. **Shin'ichi Arakawa:** Conceptualization, Resources, Supervision, Project administration, Funding acquisition. **Masayuki Murata:** Conceptualization, Resources, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Onur Alparslan reports financial support was provided by National Institute of Information and Communications Technology (NICT).

Data availability

The data that has been used is confidential.

Acknowledgments

This work is supported by the commissioned research (No. 21801) by National Institute of Information and Communications Technology (NICT), Japan. The authors thank the anonymous reviewers for their comments, which greatly improved the article.

References

[1] M.N. Ahangar, Q.Z. Ahmed, F.A. Khan, M. Hafeez, A survey of autonomous vehicles: Enabling communication technologies and challenges, Sensors 21 (3) (2021) http://dx.doi.org/10.3390/s21030706, URL https://www.mdpi.com/1424-8220/21/3/706.

[2] C.R. Storck, F. Duarte-Figueiredo, A survey of 5G technology evolution, standards, and infrastructure associated with vehicle-to-everything communications by Internet of Vehicles, IEEE Access 8 (2020) 117593–117614, http://dx.doi.org/10.1109/ACCESS.2020.3004779.

[3] D. Kombate, Wanglina, The Internet of Vehicles based on 5G communications, in: 2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data, SmartData, 2016, pp. 445–448, http://dx.doi.org/10.1109/iThings-GreenCom-CPSCom-SmartData.2016.105.

[4] IEEE Time-Sensitive Networking (TSN) task group, 2022, URL https://1.ieee802.org/tsn/.

[5] L. Deng, G. Xie, H. Liu, Y. Han, R. Li, K. Li, A survey of real-time Ethernet modeling and design methodologies: From AVB to TSN, ACM Comput. Surv. 55 (2) (2022) http://dx.doi.org/10.1145/3487330.

[6] IEEE P802.3cz multi-gigabit optical automotive Ethernet task force, 2022, URL https://ieee802.org/3/cz/index.html.

[7] H. Tsuda, R. Kubo, T. Furuya, M. Iwase, M. Morimoto, H. Kondo, Y. Amamiya, Y. Nakano, T. Tanemura, M. Murata, S. Arakawa, N. Yamamoto, A. Matsumoto, R. Takahashi, Proposal for a highly reliable in-vehicle optical network: SiPhON (Si-Photonics-Based In-Vehicle Optical Network), in: 2022 27th OptoElectronics and Communications Conference (OECC) and 2022 International Conference on Photonics in Switching and Computing, PSC, 2022, http://dx.doi.org/10.23919/OECC/PSC53152.2022.9849883.

[8] S. Jiang, Vehicle E/E Architecture and Its Adaptation to New Technical Trends, Tech. Rep., SAE Technical Paper, 2019.

[9] A.G. Mariño, F. Fons, J.M.M. Arostegui, The future roadmap of in-vehicle network processing: A HW-centric (R-)evolution, IEEE Access 10 (2022) 69223–69249, http://dx.doi.org/10.1109/ACCESS.2022.3186708.

[10] H. Askaripoor, M. Hashemi Farzaneh, A. Knoll, E/E architecture synthesis: Challenges and technologies, Electronics 11 (4) (2022) http://dx.doi.org/10.3390/electronics11040518, URL https://www.mdpi.com/2079-9292/11/4/518.

[11] T. Inoue, A. Tsuchiya, K. Kishine, D. Ito, Y. Takahashi, M. Nakamura, A burst-mode TIA with adaptive response and stable operation for in-vehicle optical networks, in: 2021 28th IEEE International Conference on Electronics, Circuits, and Systems, ICECS, 2021, pp. 1–6, http://dx.doi.org/10.1109/ICECS53924.2021.9665522.

[12] O. Alparslan, S. Arakawa, M. Murata, Next generation intra-vehicle backbone network architectures, in: Proceedings of the IEEE 22nd International Conference on High Performance Switching and Routing, HPSR, IEEE, 2021, pp. 1–7, http://dx.doi.org/10.1109/HPSR52026.2021.9481803.

[13] J. Huang, M. Zhao, Y. Zhou, C. Xing, In-vehicle networking: Protocols, challenges, and solutions, IEEE Netw. 33 (1) (2019) 92–98, http://dx.doi.org/10.1109/MNET.2018.1700448.

[14] D. Pannell, L. Chen, J. Dorr, W. Lo, D. Pannell, M. Potts, H. Zinner, A. Zu, Use cases, 2019, pp. 1–47, IEEE P802.1DG V0.4.

[15] K. Matheus, T. Königseder, Automotive Ethernet, Cambridge University Press, 2021.

[16] IEEE standard for local and metropolitan area network–bridges and bridged networks, 2018, pp. 1–1993, http://dx.doi.org/10.1109/IEEESTD.2018.8403927, IEEE Std 802.1Q-2018.

[17] IEEE standard for local and metropolitan area networks–audio video bridging (AVB) systems, 2011, pp. 1–45, http://dx.doi.org/10.1109/IEEESTD.2011.6032690, IEEE Std 802.1BA-2011.

[18] L. Zhao, P. Pop, S. Steinhorst, Quantitative performance comparison of various traffic shapers in time-sensitive networking, IEEE Trans. Netw. Serv. Manag. (2022) 1, http://dx.doi.org/10.1109/TNSM.2022.3180160.

[19] X. Jin, C. Xia, N. Guan, C. Xu, D. Li, Y. Yin, P. Zeng, Real-time scheduling of massive data in time sensitive networks with a limited number of schedule entries, IEEE Access 8 (2020) 6751–6767, http://dx.doi.org/10.1109/ACCESS.2020.2964690.

[20] E. Schweissguth, D. Timmermann, H. Parzyjegla, P. Danielis, G. Mühl, 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, 2020, pp. 1–11, http://dx.doi.org/10.1109/RTCSA50079.2020.9203662.

[21] M. Barzegaran, B. Zarrin, P. Pop, Quality-of-control-aware scheduling of communication in TSN-based fog computing platforms using constraint programming, in: 2nd Workshop on Fog Computing and the IoT, Fog-IoT 2020, Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[22] A.A. Syed, S. Ayaz, T. Leinmüller, M. Chandra, Dynamic scheduling and routing for TSN based in-vehicle networks, in: 2021 IEEE International Conference on Communications Workshops, ICC Workshops, 2021, pp. 1–6, http://dx.doi.org/10.1109/ICCWorkshops50388.2021.9473810.

[23] F. Dürr, N.G. Nayak, No-wait packet scheduling for IEEE time-sensitive networks (TSN), in: Proceedings of the 24th International Conference on Real-Time Networks and Systems, RTNS '16, Association for Computing Machinery, New York, NY, USA, 2016, pp. 203–212, http://dx.doi.org/10.1145/2997465.2997494.

[24] IEEE standard for local and metropolitan area networks–bridges and bridged networks–amendment 29: Cyclic queuing and forwarding, 2017, pp. 1–30, http://dx.doi.org/10.1109/IEEESTD.2017.7961303, IEEE 802.1Qch-2017.

[25] IEEE standard for local and metropolitan area networks–bridges and bridged networks - amendment 34: Asynchronous traffic shaping, 2020, pp. 1–151, http://dx.doi.org/10.1109/IEEESTD.2020.9253013, IEEE Std 802.1Qcr-2020.

[26] N.G. Nayak, F. Dürr, K. Rothermel, Incremental flow scheduling and routing in time-sensitive software-defined networks, IEEE Trans. Ind. Inform. 14 (5) (2018) 2066–2075, http://dx.doi.org/10.1109/TII.2017.2782235.

[27] V. Balasubramanian, M. Aloqaily, M. Reisslein, An SDN architecture for time sensitive industrial IoT, Comput. Netw. 186 (2021) 107739, http://dx.doi.org/10.1016/j.comnet.2020.107739.

[28] W. Wang, S. Yu, W. Cao, K. Guo, Review of in-vehicle optical fiber communication technology, Automot. Innov. (2022) 1–13.

[29] IEEE standard for a precision clock synchronization protocol for networked measurement and control systems, 2020, pp. 1–499, http://dx.doi.org/10.1109/IEEESTD.2020.9120376, IEEE Std 1588-2019 (Revision of IEEE Std 1588-2008).

[30] J. Muckell, P.W. Olsen, J.-H. Hwang, C.T. Lawson, S. Ravi, Compression of trajectory data: a comprehensive evaluation and new approach, GeoInformatica 18 (2014) 435–460.

[31] C. Siddique, X.J. Ban, State-dependent Self-Adaptive Sampling (SAS) method for vehicle trajectory data, Transp. Res. C 100 (2019) 224–237, http://dx.doi.org/10.1016/j.trc.2019.01.018.

[32] J. Traub, S. Breß, T. Rabl, A. Katsifodimos, V. Markl, Optimized on-demand data streaming from sensor nodes, in: Proceedings of the 2017 Symposium on Cloud Computing, SoCC '17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 586–597, http://dx.doi.org/10.1145/3127479.3131621.

[33] G. Patti, L.L. Bello, Performance assessment of the IEEE 802.1Q in automotive applications, in: 2019 AEIT International Conference of Electrical and Electronic Technologies for Automotive, AEIT AUTOMOTIVE, 2019, pp. 1–6, http://dx.doi.org/10.23919/EETA.2019.8804536.

[34] T. Steinbach, H.D. Kenfack, F. Korf, T.C. Schmidt, An extension of the OMNeT++ INET framework for simulating real-time Ethernet with high accuracy, in: Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques, SIMUTools '11, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Brussels, BEL, 2011, pp. 375–382.

[35] I. CAST, Ultra-low latency TSN networks webinar, 2021, YouTube, URL https://www.youtube.com/watch?v=_Fd6IhQLW54.

[36] B. Houtan, M. Ashjaei, M. Daneshtalab, M. Sjödin, S. Afshar, S. Mubeen, Schedulability analysis of best-effort traffic in TSN networks, in: 2021 26th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA, 2021, pp. 1–8, http://dx.doi.org/10.1109/ETFA45728.2021.9613511.