


Article

Development of Deterministic Communication for In-Vehicle Networks Based on Software-Defined Time-Sensitive Networking

Binqi Li , Yuan Zhu ^{*}, Qin Liu and Xiangxi Yao

School of Automotive Studies, Tongji University, Shanghai 201804, China; bqli_tongji@tongji.edu.cn (B.L.); 2241584@tongji.edu.cn (Q.L.); yaoxxi@tongji.edu.cn (X.Y.)

* Correspondence: yuan.zhu@tongji.edu.cn

Abstract: To support more advanced functionality in vehicles, there is the challenge of deterministic and reliable transmission of sensor data and control signals. Time-sensitive networking (TSN) is the most promising candidate to meet this demand by leveraging IEEE 802.1 ethernet standards, which include time synchronization, traffic shaping, and low-latency forwarding mechanisms. To explore the implementation of TSN for in-vehicle networks (IVN), this paper proposes a robust integer linear programming (ILP)-based scheduling model for time-sensitive data streams to mitigate the vulnerabilities of the time-aware shaper (TAS) mechanism in practice. Furthermore, we integrate this scheduling model into a software-defined time-sensitive networking (SD-TSN) architecture to automate the scheduling computations and configurations in the design phase. This SD-TSN architecture can offer a flexible and programmable approach to network management, enabling precise control over timing constraints and quality-of-service (QoS) parameters for time-sensitive traffic. Firstly, data transmission requirements are gathered by the centralized user configuration (CUC) module to acquire traffic information. Subsequently, the CNC module transfers the computed results of routing and scheduling to the YANG model for configuration delivery. Finally, automotive TSN switches can complete local configuration by parsing the received configuration messages. Through an experimental validation based on a physical platform, this study demonstrates the effectiveness of the scheduling algorithm and SD-TSN architecture in enhancing deterministic communication for in-vehicle networks.



Citation: Li, B.; Zhu, Y.; Liu, Q.; Yao, X. Development of Deterministic Communication for In-Vehicle Networks Based on Software-Defined Time-Sensitive Networking. *Machines* **2024**, *12*, 816. <https://doi.org/10.3390/machines12110816>

Academic Editors: Yuping He and Qinghui Zhou

Received: 22 October 2024
Revised: 8 November 2024
Accepted: 13 November 2024
Published: 15 November 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Keywords: time-sensitive networking (TSN); in-vehicle (IVNs); software-defined networking; deterministic communication; IEEE 802.1 Qbv; time-aware shaper (TAS)

1. Introduction

With more and more sophisticated and advanced features of connected and automated vehicles (CAVs), such as the advanced driver assistance system (ADAS) and infotainment systems, the need for high-speed, real-time, and reliable data transmission in in-vehicle networks (IVNs) has increased significantly [1]. However, conventional on-board buses, e.g., CAN, CANFD, and LIN, are no longer suitable for data transmission in these systems due to their low bandwidth. Therefore, ethernet has been introduced from the traditional IT industry to the automobile. Nevertheless, ethernet typically uses a best-effort approach to transmit data and cannot guarantee bounded-latency data transmission. To address this issue, time-sensitive networking (TSN) has been proposed by the IEEE 802.1 task group to support strict real-time requirements [2]. A novel transport protocol, TSN has been widely used in technologies such as 5G and IoT to provide low-latency communications [3,4]. TSN looks to be the backbone of next-generation automotive electronic and electrical architecture [5].

In addition to the introduction of TSN, the architecture of IVNs in autonomous vehicles has been evolving. In the earliest distributed architectures, each individual electronic control unit (ECU) was responsible for a single function, making the in-vehicle network structure complex and difficult to scale. To simplify IVNs, a domain-based architecture was

developed where all relevant functions are integrated into a single domain controller. This approach has significantly improved the integration of automotive electronic–electrical architecture (EEA) and is widely adopted today. To fully leverage the advantages of ethernet, researchers have proposed a new zonal control architecture [6]. In this architecture, the functions of each zone are managed by its own zone controller, reducing the length and weight of the vehicle’s wiring harness while enhancing the utilization of the ethernet backbone bandwidth [7].

In order to achieve bounded delay and reduce jitter during transmission, the TSN task group has proposed various traffic-shaping and -scheduling mechanisms in recent years. The time-aware shaper (TAS) mechanism proposed in the IEEE 802.1Qbv protocol [8] enables deterministic transmission of time-sensitive streams that are feasible by virtue of strict prioritization mechanisms and time-slot reservation [9]. As shown in Figure 1, traffic will be filtered into different queues according to its priority when it passes through the egress port of the switch. The gate status of each queue is controlled by the gate control list (GCL) periodically. This fine-grained time-slot control ensures that time-triggered (TT) flows that are typically used to transfer mission-critical data, such as control signals, are not disturbed by the best-effort (BE) flows. Therefore, finding a feasible GCL is crucial for achieving deterministic transmission of TT streams, even when there are BE streams transmitting simultaneously on the network at the same time.

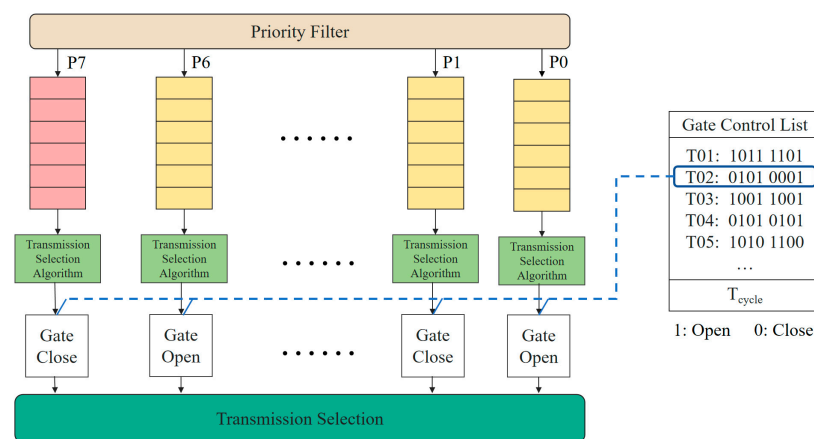


Figure 1. Time-aware shaper (TAS) mechanism in IEEE 802.1 Qbv standard.

This means that during the design phase, the network developer must plan the routing path and implement a well-calculated GCL based on information about the characteristics of all the traffic to be transported in the network, such as periodicity, load, and latency requirements. This task, referred to as Qbv scheduling, can be converted into a job-shop problem. Qbv scheduling is a job-shop problem, which is a typical combinatorial optimization problem with NP-hard difficulty [10]. In runtime, time-sensitive data must be transmitted at specified moments to align with the allocated time slots in the GCL, ensuring that time-sensitive data streams are not interfered with by other traffic. However, the TAS mechanism has limitations in practice. If, due to some uncertainty, the data miss the time slot, the delay can fluctuate significantly. Therefore, these uncertain factors must be considered in order to enhance the robustness of the system when performing Qbv scheduling.

After generating the forwarding tables and GCLs that meet the transmission requirements, network administrators must configure them to the corresponding switches in the network. However, as the topology of IVNs evolves in the direction of increasing complexity, there will also be more data streams that need to be transmitted in the future. This poses a great challenge to the traditional manual configuration approach. An emerging network management paradigm, software-defined networking (SDN) offers a flexible and efficient solution by tackling this challenge through the separation of the control plane and the data plane [11]. To guide the configuration of TSN based on SDN principles, the

IEEE 802.1Qcc standard [12,13] proposes three different types of configuration models: distributed, centralized, and fully centralized. The centralized configuration model uses employs a centralized user configuration (CUC) module to gather requirements from end systems and then passes this information on to a centralized network configuration (CNC) module. The CNC, with a global view of all switches, topology, and traffic, is responsible for configuring the network devices using remote network management protocols.

However, the majority of most current studies on TSN are limited to the simulation level and lack validation based on actual physical platforms [14]. In order to address the abovementioned challenges in the practical applications of TSN in IVNs, this paper proposes a software-defined TSN (SD-TSN) architecture to manage the routing, scheduling, and configuration tasks. Additionally, we introduce a robust integer linear programming (ILP)-based scheduling model that accounts for external influences in real-world scenarios. This algorithm aims to reduce the vulnerability of the TAS in practical applications. We integrate it into the SD-TSN to calculate transmission offsets of time-sensitive data flows on the forwarding paths.

The remainder of this paper is organized as follows. Section 2 reviews the related literature. Section 3 formulates a robust ILP-based scheduling algorithm used to compute the forwarding paths and transmission offsets. In Section 4, the workflow of the SD-TSN architecture is elaborated. An experimental validation based on a physical platform is to demonstrate the effectiveness of the robust scheduling model and SD-TSN architecture in enhancing deterministic communication for in-vehicle networks is described in Section 5. Finally, Section 6 provides conclusions and outlines future work.

2. Related Work

SAE International (Society of Automotive Engineers) has defined six levels of autonomous driving, and for level 2 and above, real-time and reliable data transmission is crucial for autonomous vehicles [15]. As a result, the evolution of E/E architecture [16] and in-vehicle time-sensitive networking [17] has garnered significant attention from researchers. Wang et al. proposed a joint routing and scheduling algorithm in order to achieve deterministic data transmission for in-vehicle networks [18]. However, the algorithm is designed for the cycle queuing and forwarding (CQF) mechanism, which is not currently supported by automotive TSN switches. To address the delay issue in in-vehicle ethernet, Chen [19] proposed a fixed-point message scheduling algorithm based on TSN technology and demonstrated its effectiveness in improving message transmission efficiency through simulation experiments.

In recent years, numerous studies have focused on the Qbv scheduling problem to compute a feasible GCL. The primary solution methods can be classified into two categories: exact methods based on the integer linear programming [20] or satisfiability modulo theories (SMT) [21,22] and optimal approximate methods based on heuristic algorithms [23] or deep learning [24]. Craciunas et al. analyzed the transmission process of TT flows in IEEE 802.1Qbv and built a constraint model for offline scheduling [25]. This constraint model has become the mainstream approach for computing the transmission offset of each time-sensitive flow along routing paths exactly. Schweissguth et al. [26] introduced a novel integer linear programming (ILP) formulation to combine the routing and scheduling process into one ILP model. Compared to separate routing and scheduling approaches, this joint method offers a larger solution space, which improves the success rate of scheduling. In addition, scholars have optimized the solution process of this scheduling problem. A conflict-aware stream partitioning technique is proposed to improve the scalability of scheduling and can be used to select the optimal path from multiple reachable paths [27]. Kwon [28] applied machine learning to optimize the traffic scheduling problem in autonomous vehicle networks. The experimental results demonstrated the feasibility of optimizing TSN traffic scheduling using artificial intelligence-based algorithms. To compare the performance of various scheduling algorithms, Stüber implemented eleven algorithms and evaluated them with respect to schedule quality and runtime [29].

Configuring TSN based on SDN principles is also a major point of interest for researchers. Based on the centralized configuration model, Luo Kun [30] designed a TSN configuration management system that supports topology discovery, scheduling computation and configuration functions. However, this system is resource-intensive and requires end systems and switches to support advanced protocols, such as the link layer discovery protocol (LLDP). Consequently, deploying this system on resource-constrained platforms, such as automotive embedded systems, is challenging. To address the need for dynamic reconfiguration in industrial Internet of Things scenarios, Venkatraman et al. integrated routing and scheduling algorithms into an SDN controller to calculate TSN configurations online [31]. Junli et al. conducted a simulation and concluded that SDN-based TSN configurations can meet the end-to-end latency requirements of time-sensitive data streams [32]. Dürr developed an SDN-based TSN framework integrating IEEE 802.1Qbv and IEEE 802.1Qcc standards to reduce the cost of validation for TSN deployments [33].

3. Problem Formulation of TAS Scheduling

3.1. System Model

An example of an in-vehicle network topology with a zonal architecture is illustrated in Figure 2. This topology consists of four different types of nodes: endpoints, switches, gateway, and PC. Endpoints represent components such as microcontrollers, sensors, or actuators in the vehicle. They are directly connected to switches within their respective zone via ethernet. All switches are interconnected through a gateway, which in turn is connected to a PC, representing a high-performance computing unit. The subsequent construction of the physical experimental platform is based on this topology, and the two data flows (Figure 2) will be discussed in more detail in Section 4.2. A directed graph model can be used to represent the network topology, capturing both the structure of the network and the attributes of nodes and links within it.

$$\begin{cases} G = (V, E); \\ v = (v.name, v.ip, v.mac); \\ e = (v_s, v_d, p_s, p_d, e.bd), \end{cases} \quad (1)$$

Here, G represents the entire topology model, V is the vertex set consisting of all nodes within the topology, and E is the set of all directed edges in the topology. Each vertex $v \in V$ is defined by a three-tuple to store the node information, where $v.name$, $v.ip$, $v.mac$ denote the name, IP, address, and MAC address, respectively, of the node. Each edge $e \in E$ is defined by a five-tuple to store the edge attributes, where v_s, v_d are the source vertex and destination vertex of the directed edge, p_s, p_d represents which port of the source node v_s the edge is concatenated from and which port of the destination node v_d it is connected to, and $e.bd$ is the bandwidth of the edge.

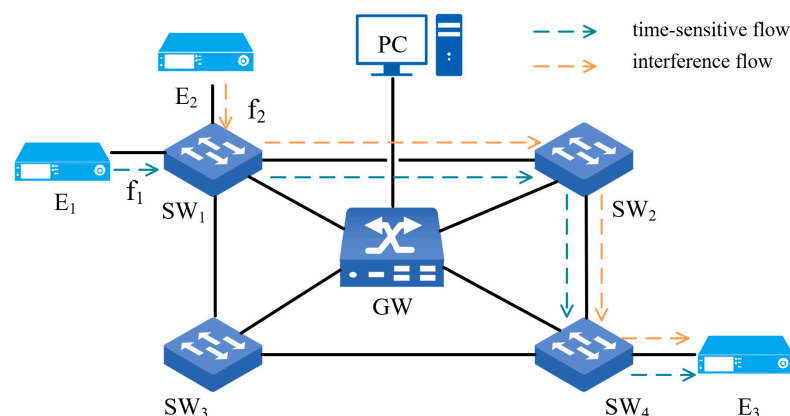


Figure 2. An automotive zonal architecture of in-vehicle networks.

To ensure deterministic transmission of time-sensitive data flows f , it is essential to gather information on their attributes. Subsequently, the collection F of these modeled attributes serves as input for the subsequent scheduling algorithm. A detailed definition of the traffic model f is provided below.

$$f = (f.src, f.dst, f.T, f.priority, f.load, f.latency, f.vid). \quad (2)$$

The symbols in the equation represent the source node, destination node, period priority, payload, maximum latency requirement, and VLAN ID, respectively.

3.2. Formulation of the Robust Scheduling Model

Scheduling for time-sensitive data flows is a classical job-shop problem (JSP) that which is NP-complete, and is recognized as one of the most challenging combinatorial optimization problems [34]. Integer linear programming (ILP) is widely acknowledged as an effective approach for tackling this issue. In this subsection, a robust ILP-based scheduling model is formulated that can be applied to practical in-vehicle network scenarios. To reduce computational complexity, this study segregates separates routing and scheduling computations, utilizing the basic shortest-path algorithm to determine the forwarding paths of time-sensitive data flows.

3.2.1. Transmission Start Constraint

The start of the transmission time on each link plus the transmission delay must fit within the flow period to ensure that sufficient time remains for transmission.

$$\forall f_k \in F, \forall e_m \in R_k : \quad (3)$$

$$0 \leq t_k^m \leq f_k.T - \frac{f_k.load}{e_m.bd},$$

Here, R_k represents the routing paths of f_k . The decision variable t_k^m represents the transmission offset of flow f_k on edge e_m .

3.2.2. Flow Isolation Constraint

In practice, a data flow may consist of one or multiple frames. Instead of employing frame isolation constraints, which offer higher fault tolerance, this paper opts for flow isolation constraints to reduce the number of decision variables and thereby simplify the scheduling model. As shown in Figure 3, assume that two flows, flow k and flow l , are passing through a common edge e_m . A constraint must be placed on the timing of their arrival at this edge. If flow k arrives at e_m first, flow l must wait until flow k completes its transmission on the common edge e_m before it can proceed to the next edge e_m from the previous hop edge e_q . The same applies if flow l arrives at e_m first. Specifically, at any given time, a switch output port's buffer queue of the switch's egress port is restricted to storage frames exclusively from the same data flow. Only after the current flow has been completely transmitted from the port will the next flow be transmitted from its previous node to the current node.

$$\forall f_k, f_l \in F | k \neq l, \forall e_m \in (R_k \cap R_l) | e_m.v_s \neq (f_k.src \cup f_l.src),$$

$$\forall \mu \in \left[0, \frac{hp_{kl}}{f_k.T}\right), \forall \nu \in \left[0, \frac{hp_{kl}}{f_l.T}\right) : \quad (4)$$

$$\left(t_k^m + \mu * f_k.T + \frac{f_k.load}{e_m.bd} \leq t_l^q + \nu * f_l.T\right) \vee \left(t_l^m + \nu * f_l.T + \frac{f_l.load}{e_m.bd} \leq t_k^p + \mu * f_k.T\right),$$

Here, hp_{kl} represents the hyper-period between flows f_k and f_l , which equals the least common multiple of their respective periods. An edge e_m is a common edge obtained by taking the intersection of the routing paths R_k and R_l of two flows. Variables μ and ν denote the instances of flows f_k and f_l within this hyper-period. The indices p and q signify the respective previous-hop link edges for flows f_k and f_l before arriving at link edge e_m .

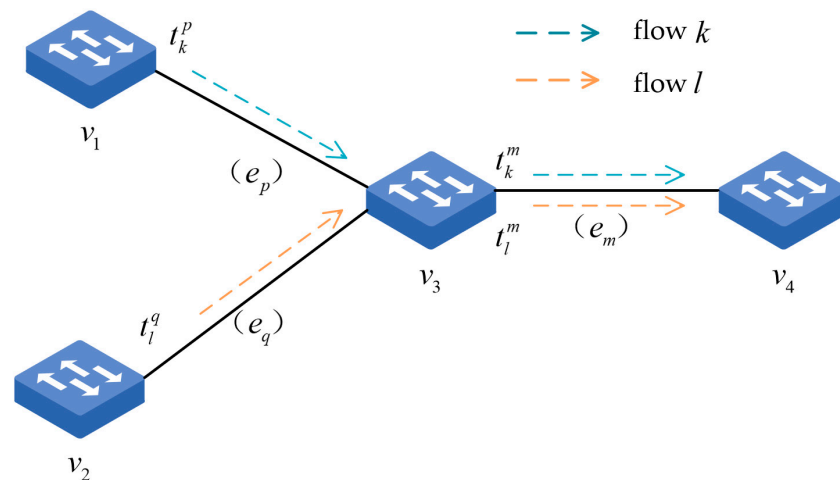


Figure 3. Flow isolation of two flows on the common edge through which they pass.

3.2.3. Link Resource Constraint

The time slots assigned to all flows routed through the same link must not overlap temporally in the time dimension to ensure conflict-free transmission. Specifically, for every pair of flows f_k and f_l , the time slot allocated to f_k must either precede or succeed that allocated to f_l .

$$\begin{aligned} &\forall f_k, f_l \in F | k \neq l, \forall e_m \in (R_k \cap R_l); \\ &\forall \mu \in \left[0, \frac{hp_{kl}}{f_k.T}\right), \forall \nu \in \left[0, \frac{hp_{kl}}{f_l.T}\right): \\ &\left(t_k^m + \mu * f_k.T + \frac{f_k.load}{e_m.bd} + GB + C \leq t_l^m + \nu * f_l.T - C\right) \vee \\ &\left(t_l^m + \nu * f_l.T + \frac{f_l.load}{e_m.bd} + GB + C \leq t_k^m + \mu * f_k.T - C\right), \end{aligned} \tag{5}$$

Here, GB represents the guard band, defined as the duration required to transmit a maximum-sized frame over the link. In addition to the guard band, we introduce a compensation value C within this constraint to increase the separation between two flows. As illustrated in Figure 4, this allows us to add compensation at both ends of each flow’s required transmission slot, thereby mitigating the potential impact of external factors that may cause actual traffic to arrive earlier or later than expected. This enhancement improves the interference resilience of the TAS mechanism when applied in practical scenarios.

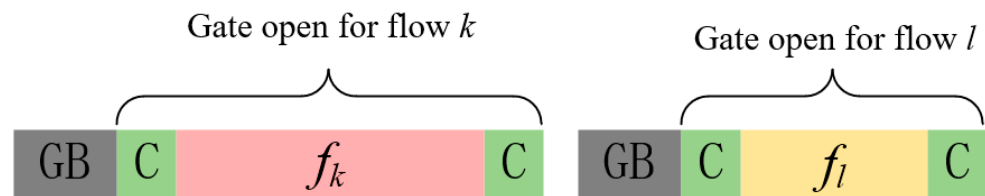


Figure 4. Guard band and compensation in GCL.

3.2.4. Flow Transmission Constraint

When a flow originates from its source and traverses the routing path towards its destination, it incurs delays due to factors such as processing delay, propagation delay, and transmission delay. In the context of in-vehicle network topology, where wire lengths typically do not exceed 5 m, the propagation delay through each link is negligible, at the nanosecond scale, and is therefore not considered in this study.

$$\begin{aligned} &\forall f_k \in F, \forall e_m, e_n \in R_k | e_n.v_s == e_m.v_d : \\ &t_k^m + \frac{f_k.load}{e_m.bd} + d_{proc} \leq t_k^n, \end{aligned} \tag{6}$$

Here, d_{proc} represents the processing delay within the switch. This constraint stipulates that a flow can only be scheduled on the succeeding edge e_n after completing transmission on the preceding edge e_m , accounting for switch processing delays.

3.2.5. End-to-End Latency Constraint

Finally, we enforce constraints by comparing the departure time from the source node with the transmission time at the link where the destination node is located, ensuring that the end-to-end latency requirements are met for time-sensitive data flows.

$$\begin{aligned} \forall f_k \in F, e_{first} \in R_k \mid e_{first}.v_s == f_k.src, \\ e_{last} \in R_k \mid e_{last}.v_d == f_k.dst : \\ t_k^{last} + \frac{f_k.load}{e_{last}.bd} - t_k^{first} \leq f_k.latency, \end{aligned} \tag{7}$$

Here, e_{last} represents the last edge before the destination node and e_{first} represents the first edge after the source node.

4. Software-Defined TSN Architecture

In order to achieve real-time transmission of critical data streams in vehicular time-sensitive networking, intricate complex configurations involving the routing and scheduling of switches are essential. Given the shortcomings of static configuration methods characterized by high workload and low flexibility, we propose a centralized configuration architecture (Figure 5) based on SDN principles. This architecture aims to dynamically control the forwarding path and reserve bandwidth in in-vehicle time-sensitive networks to meet the stringent deterministic transmission demands requirements of critical data streams.

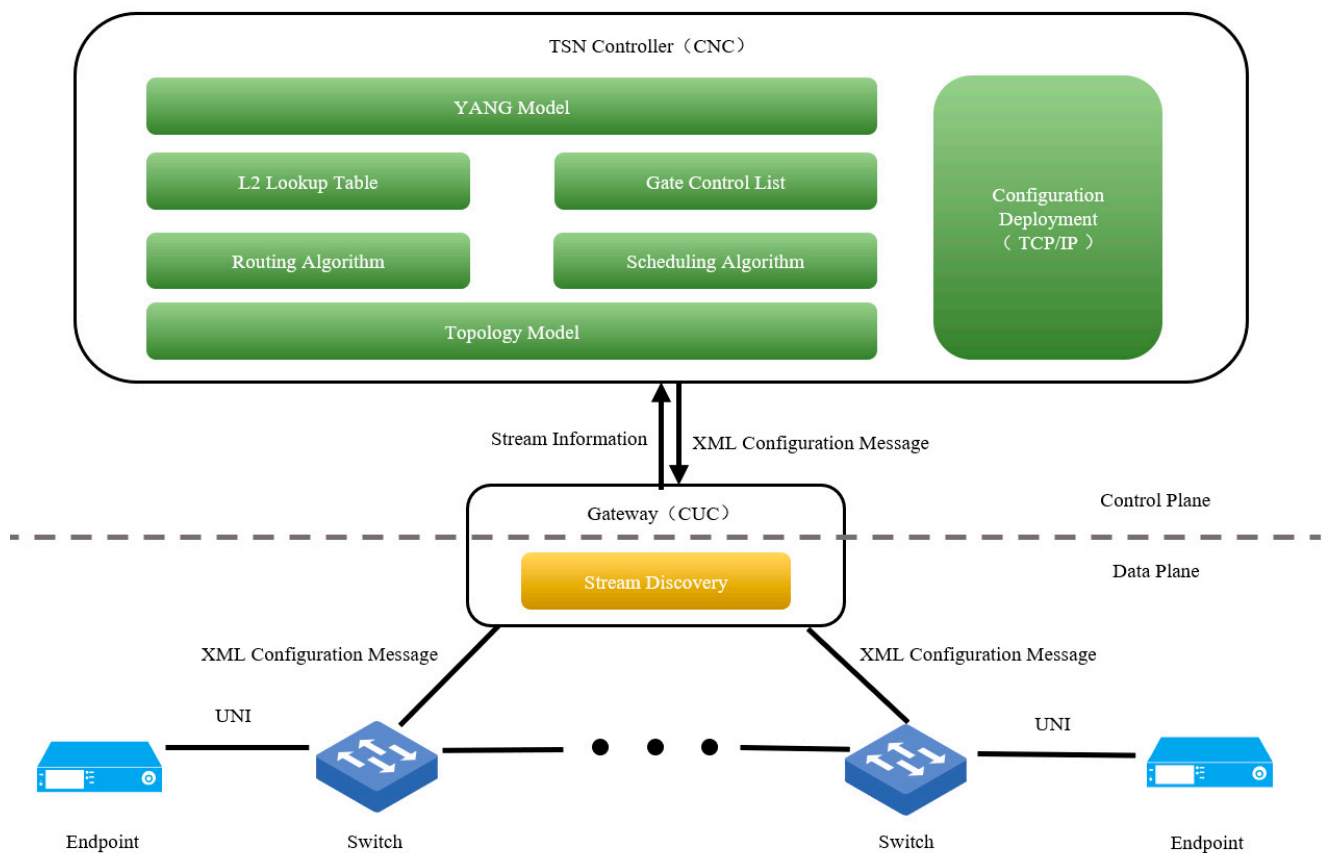


Figure 5. Software-defined TSN (SD-TSN) architecture for in-vehicle networks.

4.1. Working Principle of the SD-TSN Mechanism

As depicted in Figure 5, the SD-TSN mechanism operates in vehicular networks by coordinating interactions among four entities—end nodes, switches, gateways, and controllers—facilitating the execution of the time-aware shaper (TAS) TAS. The detailed operational principles are elaborated as follows.

1. Achieving precise clock synchronization across the entire network using the precise time protocol (PTP). In time-sensitive networks, a unified clock reference forming the cornerstone is essential for numerous traffic shaping and scheduling mechanisms.
2. Talkers or listeners on endpoints interact with the CUC module through the user-network interface (UNI), transmitting diverse stream attribute information. The purpose of this interaction process is to request network resources from the CUC to fulfill their transmission requirements.
3. The gateway incorporates includes the CUC module, which is responsible for the stream discovery process. It establishes connections to switches to gather real-time traffic information. After that, the collected stream information is relayed to the CNC module.
4. Upon receiving stream information, the TSN controller, acting as the CNC, performs routing and scheduling calculations based on the global topology model. It generates L2 lookup tables and gate control lists from these calculations and encodes and transfers these to YANG models for switch configuration. The configuration messages are then sent to TSN switches in XML format via TCP/IP connections.
5. The TSN switch decodes the XML messages received, extracts configuration details, and invokes reserved APIs within the chip driver to finalize the configuration of the local L2 lookup table and gate control list within the TAS mechanism.

4.2. Stream Discovery and Information Collection

In the SDN architecture, the CUC utilizes the UNI for network-wide stream discovery and transmits the gathered traffic information to the CNC. To enhance traffic information collection, the proposed centralized configuration model situates the CUC within the in-vehicle gateway. Since the gateway directly interfaces with all switches in the network, talkers and listeners can engage with the CUC via the neighboring switch to exchange traffic details. It is important to state note that the TSN entities (talker and listener) can choose the appropriate communication protocol to interact with the CUC based on the upper layer application. For example, SOME/IP and data distribution services (DDSs) are commonly used as middleware for implementing service-oriented communication (SOC) in vehicle ethernet. Their dynamic discovery mechanisms are well suited for implementing stream discovery.

4.3. Path Control and L2 Lookup Table Configuration

Before scheduling time-sensitive flows using the TAS mechanism, we need to identify the switch ports they traverse within the network. In other words, the routing paths of these flows must be explicitly controlled to comply with the GCL within the TAS. To eliminate the uncertainty introduced by the self-learning forwarding method, the L2 lookup table on each switch must be configured during the design phase.

In this paper, we adopt a centralized approach based on our proposed SD-TSN architecture to cope with these time-consuming configuration tasks. The routing algorithm is deployed within the TSN controller so that it can adaptively compute the forwarding paths based on prior knowledge of traffic and topology. Scholars have conducted extensive research on routing algorithms and proposed a variety of routing methods from different perspectives, such as shortest-path algorithms, load-balancing-aware routing algorithms, and multipath routing algorithms. Since this work focuses on the implementation process of path control based on the SDN and does not explore the performance of different routing algorithms, we employed the shortest-path algorithm, which is the most commonly used approach, in subsequent experiments.

The L2 lookup table output generated by the routing algorithm is stored in the YANG model and converted into an XML-formatted message to be distributed to the TSN switch. This paper defines the YANG model shown in Table 1 for configuring L2 lookup table entries on TSN switches, specifying mandatory attributes such as the index number, destination MAC address, source port, VLAN ID, and destination port information.

Table 1. The YANG model for L2 lookup table configuration.

Entry	Entry Type	Value Type	Description
index	leaf	Uint16	The index number of this entry in L2 lookup table.
macaddr	leaf	String	The destination MAC address of this stream.
srcport	leaf	Uint16	The source port through which this stream enters the switch.
vlanid	leaf	Uint16	The VLAN id used by this stream.
destport	leaf	Uint16	The destination port through which this stream departs the switch.

4.4. Bandwidth Reservation and GCL Configuration

The purpose of bandwidth reservation is to ensure that time-sensitive data streams have exclusive access to bandwidth resources regardless of the circumstances. In order to accomplish the reservation of bandwidth reservation in in-vehicle networks, the time slots exclusive to time-sensitive streams in the GCL must be designed first, followed by the GCL configurations on the switches. The ILP scheduling algorithm formulated in Section 2 is deployed on the TSN controller for computing a GCL that satisfies meets the transmission requirements sent by the CUC.

To achieve bandwidth reservation based on time-aware shaping, we define a YANG model, shown in Table 2, for GCL configuration. This YANG model specifies which port on the switch to be configure initially. Subsequently, a list format stores the TAS scheduling entries, where each entry includes its index, the status of queue gate controls, and the duration for which the gate status will be maintained. Once the configuration is complete, the TAS iteratively executes the entries in GCL to perform real-time scheduling. It is important to note that the scheduling model proposed in Section 2 outputs only the initial time offsets for each flow at the start of transmission along their forwarding paths. To calculate the complete configuration of gate control lists based on these timings, we introduce the GCL generation algorithm depicted shown in Algorithm 1.

Table 2. The YANG model for TAS configuration.

Entry	Entry Type	Value Type	Description	
port	leaf	Uint16	The port number to be configured on the switch.	
tasScheduleEntry (list)	index	leaf	Uint16	The index number of this schedule entry in GCL.
	triggerTime	leaf	Uint32	The duration of this schedule entry.
	gateStatus	leaf	Uint8	The gate status for each queue of this schedule entry.
tasParameters (container)	gateStatus	leaf	Uint8	The initial gate status before TAS startup.
	controlListLength	leaf	Uint16	The number of GCL scheduling entries.
	cycleTime	leaf	Uint32	The cycle time of the whole GCL.
	cycleTimeExtension	leaf	Uint32	The extension time when the TAS is being reconfigured.
BaseTime	container	Uint32	The time of TAS startup.	

This algorithm takes the network topology G , flow set F , and scheduling result T as inputs, and outputs YANG model information for GCL configurations. Current mainstream scheduling algorithms primarily focus on periodic time-sensitive (TS) data streams for scheduling. To simplify the scheduling model complexity, the output result T contains only the transmission offset of flow f on path e within the first cycle. However, the

GCL cycle on switches throughout the network must remain consistent and equal to the least common multiple of all periodic time-sensitive flow cycles, referred to as the hyper-period. Algorithm 1 calculates this hyper-period hp in lines 1–3. Then, lines 4–11 of the algorithm compute the time points O when gates with priority 7 are open for all TS traffic across the entire hyper-period. Once these time points are determined for each path, GCL configurations are generated based on the points in $O[e]$. The algorithm takes into account varying bandwidth capacities across links in the in-vehicle network and computes the required durations for gate openings and guard bands for each link. Since compensation has already been accounted for within the link resource constraint, the compensation values are incorporated into the duration calculations in line 18. Subsequently, the algorithm proceeds in a loop: lines 19–21 add schedule entries for non-time-sensitive traffic; lines 22–24 add guard band entries; and lines 25–27 add entries for time-sensitive traffic. The number of cyclic executions of the code is proportional to the total number of flows, assuming the number of edges in the topology and the number of instances of each flow in the hyper-period are constant. As a result, the complexity of Algorithm 1 is linear and feasible for practical implementation.

Algorithm 1 GCL Generation Algorithm

```

Input:  $G, F, T$ 
Output:  $O, YANG$ 
1:  for  $f \in F$  do
2:     $hp = \text{Lcm}(hp, f, T)$ ;
3:  end for
4:  for  $f \in F$  do
5:     $n = hp / f.T$ ;
6:    for  $T[f][e] \in T[f]$  do
7:      for  $i \in n$  do
8:         $O[e].\text{insert}(T[f][e] + i * f.T)$ ;
9:      end for
10:   end for
11: end for
12: for  $e \in E$  do
13:    $YANG.\text{port} = \text{LookupPort}(e.v_s, v_d)$ ;
14:    $\text{accuLen} = 0$ ;
15:    $i = 0$ ;
16:   for  $t \in O[e]$  do
17:      $GB = \text{MaxFrame} / e.bd$ ;
18:      $\text{duration} = f.\text{load} / e.bd + 2 * C$ ;
19:      $YANG_i.\text{index} = i++$ ;
20:      $YANG_i.tt = t - \text{accuLen} - GB - C$ ;
21:      $YANG_i.gs = 01111111$ ; /*schedule entry for non-TS traffic*/
22:      $YANG_i.\text{index} = i++$ ;
23:      $YANG_i.tt = GB$ ;
24:      $YANG_i.gs = 00000000$ ; /*guard band entry*/
25:      $YANG_i.\text{index} = i++$ ;
26:      $YANG_i.tt = \text{duration}$ ;
27:      $YANG_i.gs = 10000000$ ; /*schedule entry for TS traffic*/
28:      $\text{accuLen} = t + \text{duration} + C$ ;
29:   end for
30: end for
31: Return  $YANG$ 

```

Upon completion of YANG generation for GCL configurations, the model is converted to XML format and deployed to TSN switches via a TCP connection, thereby enabling bandwidth reservation based on GCL.

5. Experimental Evaluation

In this section, an experimental evaluation is presented to validate the effectiveness of the proposed scheduling algorithm and SD-TSN mechanism in ensuring deterministic communication.

5.1. Experimental Platform

To simulate a realistic in-vehicle network scenario as closely as possible, we developed a physical validation platform for automotive ethernet, as shown in Figure 6. This platform features essential components, including two Broadcom (California, USA) Raspberry Pi 5 boards, four NXP (Eindhoven, The Netherlands) SJA1110 switches, two NXP S32G2 evaluation boards, and a PC (connected to the central S32G2 via the yellow ethernet cable). The NXP SJA1110 is an automotive ethernet switch that supports some basic TSN protocols, such as IEEE 802.1AS [35] and IEEE 802.1Qbv [34]. On the other hand, the S32G2, a specialized automotive processor chip developed by NXP, is designed for future in-vehicle computing and networking architectures. Its robust computational capabilities render it suitable for deployment as an in-vehicle gateway.

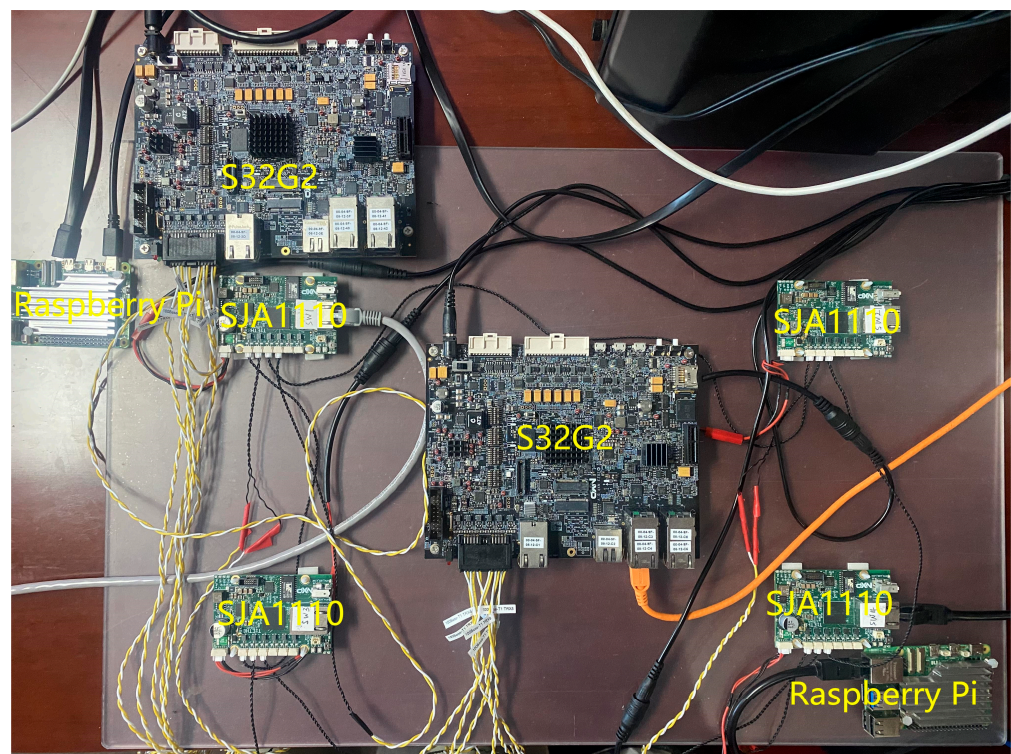


Figure 6. The experimental platform for simulating in-vehicle networks.

This experimental platform is constructed in accordance with the zonal architecture topology shown in Figure 2. In this configuration, the SJA1110 devices function as TSN switches, two Raspberry Pi 5 boards operate as endpoints E_1 and E_3 , the leftmost S32G2 serves as endpoint E_2 , and the central S32G2 acts as the gateway, interfacing with the PC that hosts the TSN controller. All nodes in the platform are interconnected via 100Base-T1 onboard ethernet with a bandwidth of 100 Mbps.

To simulate the whole working process of the SD-TSN mechanism, we chose to deploy DDS applications at both endpoints and gateways, where they function to serve as data producers and consumers in the application layer. What is more, this setup allows the gateway's CUC to utilize DDS discovery messages for stream discovery functionality. A TSN controller is deployed on the PC to act as the CNC module, executing routing and scheduling algorithms, along with the configuration algorithm proposed in this paper. The network topology is modeled by the igrph package, which is a free and open-source package for network analysis. The scheduling model is built and solved using a Gurobi optimizer, which is a professional solver for ILP optimization problems. To deploy the configurations, the PC connects to the gateway via ethernet and transmits configuration messages over a TCP/IP connection. Since the SJA1110 does not support the NETCONF

network management protocol in SJA1110, we developed parsing capabilities within YANG models in its software stack. This enables the switch to dynamically invoke interfaces from its SDK to configure the L2 lookup table and TAS, ensuring the transmission of time-sensitive data.

5.2. Experimental Implementation

During the research, two data flows, as shown in Figure 2, were designed to simulate the working conditions where a mission-critical stream is transmitted simultaneously with a background stream. Flow 1 represents the time-sensitive flow and flow 2 represents the non-time-sensitive interference flow. FastDDS in FastDDS is deployed at the application layer to construct these two flows, and the attributes of these two flows are configured according to Table 3.

Table 3. Experimental flow attributes.

Name	Talker	Listener	Priority	Period/ms	Payload/Byte
Flow 1	E ₁	E ₃	7	50	1024
Flow 2	E ₂	E ₃	0	10	3200~102,400

Flow 1 represents a time-sensitive data stream of highest priority, while stream flow 2 serves as an interfering stream with lower priority and a variable payload size to simulate the state of the network from idle to busy. In the experiments, the maximum allowable latency for flow 1 is set to 500 microseconds. The talker for flow 1 is deployed at endpoint E₁, and the listener is positioned at endpoint E₃. The routing path of flow 1 is established as E₁–SW₁–SW₂–SW₄–E₃. The talker for flow 2 is deployed at endpoint E₂, and the listener is positioned at endpoint E₃ as well. To ensure that flow 2 interferes with the transmission of flow 1, the routing path is set to E₂–SW₁–SW₂–SW₄–E₃.

5.3. Experimental Results

First of all, precise clock synchronization was established across the entire experimental platform, achieving sub-microsecond synchronization accuracy between adjacent nodes. Leveraging this setup, the end-to-end latency of flow 1 was measured by timestamping frames at both the sender and receiver ends. The experiment recorded the end-to-end latency of 100 consecutive transmissions of flow 1 with under two different scheduling mechanisms: the traditional strict priority (SP) scheduling mechanism, where TSN is not required, and the TAS mechanism, which is facilitated by the SD-TSN architecture proposed in this paper. We compared the latency and jitter of the time-sensitive streams under varying degrees of interference to assess their impact on deterministic transmission.

The latency test results of the time-sensitive data flow as the load of interfering traffic increases from idle to full are shown in Figure 7. When the load of interfering traffic is low, both the SP and TAS mechanisms exhibit relatively small and stable end-to-end delays. However, when the interfering traffic load exceeds 12,800 bytes, the end-to-end latency under strict priority scheduling deteriorates significantly, exhibiting not only increased latency but also greater jitter. The reason for this is that while the TT flow is prioritized for transmission, it must wait for the transmission of interfering traffic frames to complete transmission before it can acquire transmission rights. This introduces additional and unpredictable queuing delays for the TT flow that are unpredictable, thereby inevitably contributing to increased jitter and end-to-end latency.

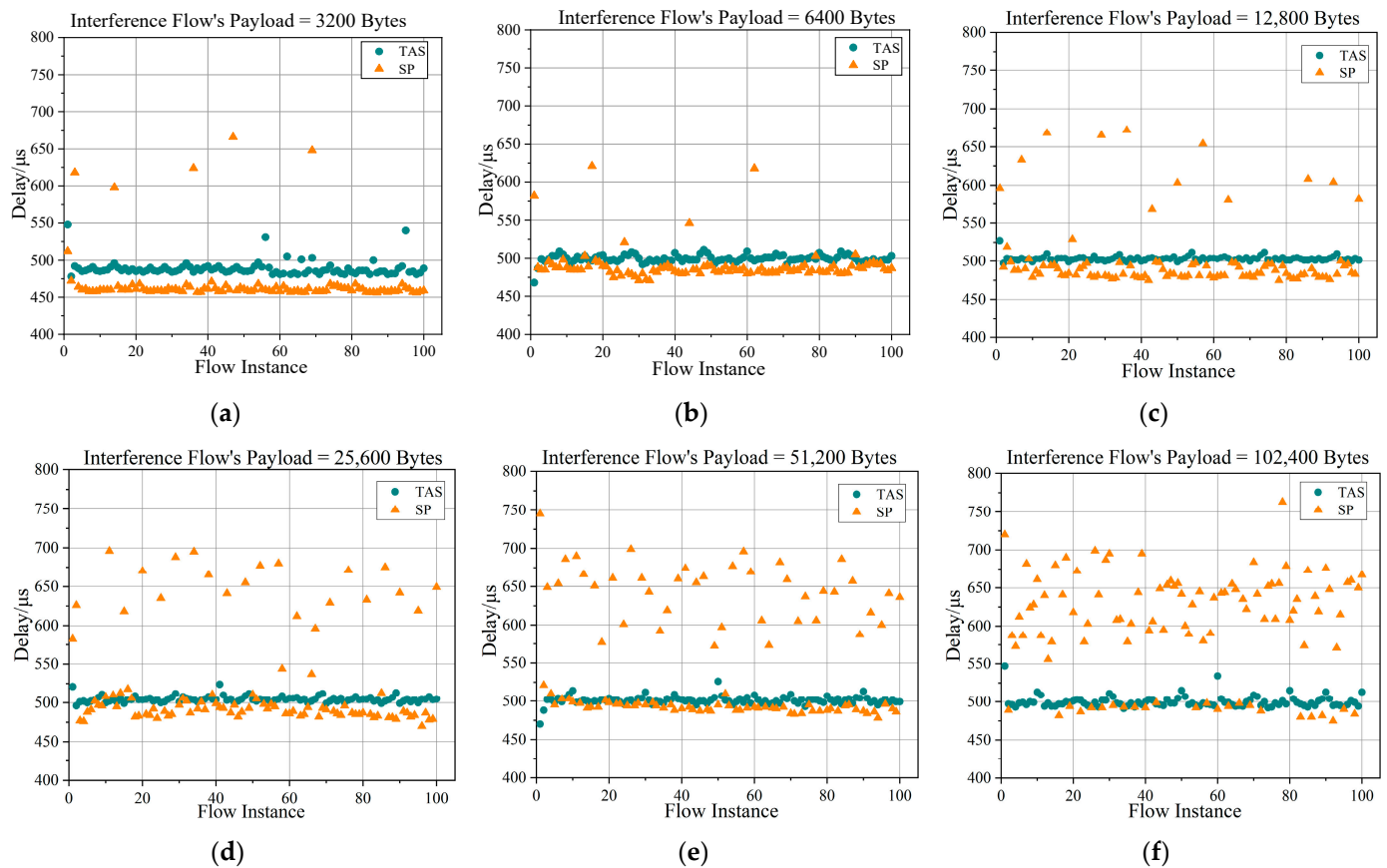


Figure 7. The end-to-end latency of TT flow instances under different degrees of interference. (a) Interference flow at 2.56 Mbps; (b) interference flow at 5.12 Mbps; (c) interference flow at 10.24 Mbps; (d) interference flow at 20.48 Mbps; (e) interference flow at 40.96 Mbps; (f) interference flow at 81.92 Mbps.

In comparison to SP scheduling, it is evident that after adopting the SD-TSN mechanism, the TAS ensures that the end-to-end latency of the TT flow remains unaffected, regardless of variations in the load of interfering traffic. This is attributed to our proposed TAS scheduling model, which allocates dedicated transmission slots for the TT flow and ensures that its end-to-end latency requirements are met through constrained scheduling.

To further demonstrate the effectiveness of our study approach in achieving deterministic transmission, Figure 8 illustrates the latency distribution under varying levels of interference traffic, represented by boxplots. It is evident that as the intensity of interference increases, the average latency under the SP mechanism shows an upward trend, accompanied by a broader overall distribution range. In contrast, with the implementation of TAS, the latency distribution of the TT time-sensitive flow stabilizes around 500 microseconds with minimal fluctuation, indicating that compliance with the requirements for deterministic transmission are consistently met.

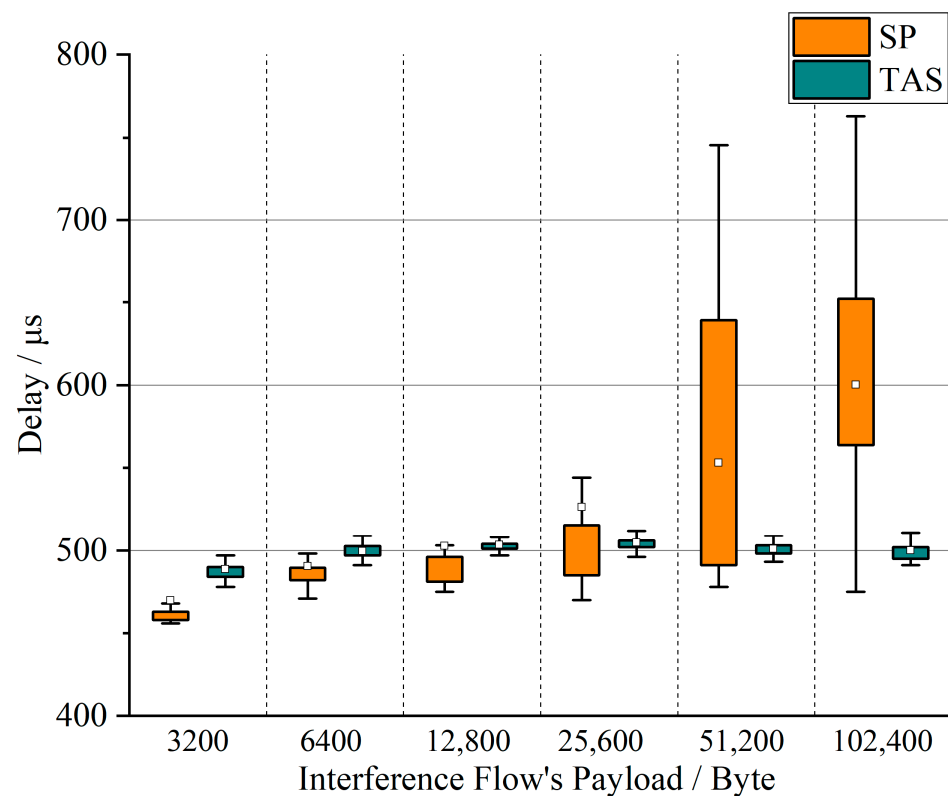


Figure 8. Latency distribution across different interference traffic loads.

6. Conclusions

Time-sensitive networking (TSN) is poised to become the backbone network in the next generation of automotive E/E architecture. This paper investigated the practical applications of time-sensitive networking for deterministic communication in autonomous vehicles. Specifically, we propose a robust ILP-based scheduling model designed to tolerate the uncertainties encountered in real-world working conditions. To address the complexities of TSN configuration in practical applications, an SD-TSN architecture is proposed to dynamically automate the configuration process for in-vehicle TSN networks. We formulated a robust ILP-based scheduling model for the TAS mechanism to achieve deterministic transmission for time-sensitive data flows. What is more, the YANG models for configuring L2 lookup tables and the GCL were established to implement path control and bandwidth reservation. The experiments conducted on a physical platform demonstrated that the SD-TSN mechanism can ensure that other traffic does not interfere with time-sensitive flows, effectively guaranteeing stringent bounded latency and jitter. The compensation values in the robust scheduling algorithm are currently determined based on experience. In future work, we will further explore the relationship between this compensation value and factors such as the size of the data load, the processing power, and the operating system of the end system so as to propose a method that can adaptively calculate the compensation value in the scheduling algorithm. The effectiveness of the scheduling model and the SD-TSN architecture was evaluated using a physical platform. Based on the experimental results, the following key conclusions can be drawn.

1. The robust ILP-based scheduling model effectively mitigates the vulnerability of the TAS mechanism and improves the availability of the TAS under real-world operating conditions.
2. The computation and configuration of the forwarding table and gate control list can be automatically completed by the SD-TSN architecture, significantly reducing the workload of network developers and minimizing errors associated with manual configuration.

- The application of the TAS mechanism in IVNs is practical. Experimental tests based on physical platforms demonstrate that the TAS offers significant advantages in ensuring bounded delay and jitter, outperforming strict priority scheduling.

This study makes a noteworthy contribution to promoting the practical application of TSN in the automotive domain. Automotive developers can leverage the concepts presented in this paper to achieve deterministic transmission of critical data and signals in vehicles, which is essential for functionality of modern automobiles.

It is important to acknowledge that TSN still has a long way to go before it can be fully commercialized in the automotive industry. Further research is needed to enhance the reliability of TSN in in-vehicle networks. Specifically, when streams miss the specified gating, a key challenge is how to prevent subsequent streams from being affected and how to recover the scheduling. This is a critical issue that must be addressed in future research.

Author Contributions: Conceptualization, B.L. and Y.Z.; methodology, B.L.; software, B.L.; validation, X.Y. and Q.L.; data curation, X.Y.; writing—original draft preparation, B.L.; writing—review and editing, Y.Z.; visualization, Q.L.; supervision, Y.Z.; project administration, Y.Z.; funding acquisition, Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Perspective Study Funding of Nanchang Automotive Institute of Intelligence and New Energy, Tongji University (grant TPD-TC202211-07) and the APC was funded by the Research on Computing Power Sharing Mechanism of In-Vehicle Network Based on DDS and AVTP of UAES Advanced Research Project (NE-2023-6V2).

Data Availability Statement: The original data presented in the study are openly available on Github at <https://github.com/bqli1996/Machines.git>, accessed on 22 October 2024.

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

Abbreviations

SD-TSN	software-defined time-sensitive networking
IVNs	in-vehicle networks
TAS	time-aware shaper
GCL	gate control list
SDN	software-defined networking
YANG	Yet Another Next Generation
CUC	centralized user configuration
CNC	centralized network configuration
PTP	precision time protocol
QoS	quality of service
ILP	integer linear programming
TT	time-triggered
BE	best effort
SP	strict priority
DDS	data distribution service
SOME/IP	service-oriented middleware over IP
UNI	user-network interface
XML	extensible markup language

References

- Zhang, C.; Zhou, W.; Yin, Y.; Li, Z.; Gong, J.; Zhang, K. Deterministic communications for in-vehicle network: Overview and challenges. In Proceedings of the 2021 2nd International Conference on Artificial Intelligence and Information Systems, Chongqing, China, 28–30 May 2021; pp. 1–6.
- Deng, L.; Xie, G.; Liu, H.; Han, Y.; Li, R.; Li, K. A survey of real-time ethernet modeling and design methodologies: From AVB to TSN. *ACM Comput. Surv. (CSUR)* **2022**, *55*, 1–36. [[CrossRef](#)]
- Muslim, A.B.; RTönjes, R.; Bauschert, T. Synchronizing TSN Devices via 802.1AS over 5G Networks. *Electronics* **2024**, *13*, 768. [[CrossRef](#)]

4. Fletcher, E.P.M.; Ridge, D.; Michaels, A.J. Low-Latency Wireless Network Extension for Industrial Internet of Things. *Sensors* **2024**, *24*, 2113. [[CrossRef](#)] [[PubMed](#)]
5. Alparslan, O.; Arakawa, S.I.; Murata, M. Next generation intra-vehicle backbone network architectures. In Proceedings of the 2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR), Paris, France, 7–10 June 2021; pp. 1–7.
6. Bornemann, M. Zone Controllers Build Bridge to Tomorrow's Technology, Aptiv White Paper. 2021. Available online: https://www.aptiv.com/docs/default-source/white-papers/2021_aptiv_whitepaper_zonecontroller.pdf (accessed on 10 October 2024).
7. Park, C.; Park, S. Performance Evaluation of Zone-Based In-Vehicle Network Architecture for Autonomous Vehicles. *Sensors* **2023**, *23*, 669. [[CrossRef](#)] [[PubMed](#)]
8. IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 25: Enhancements for Scheduled Traffic, IEEE Standard 802.1Qbv-2015. 2016. Available online: <https://ieeexplore.ieee.org/document/8613095> (accessed on 22 October 2024).
9. Stüber, T.; Osswald, L.; Lindner, S.; Menth, M. A survey of scheduling algorithms for the time-aware shaper in time-sensitive networking (TSN). *IEEE Access* **2023**, *11*, 61192–61233. [[CrossRef](#)]
10. Brinkkötter, W.; Brucker, P. Solving open benchmark instances for the job-shop problem by parallel head–tail adjustments. *J. Sched.* **2001**, *4*, 53–64. [[CrossRef](#)]
11. Feamster, N.; Rexford, J.; Zegura, E. The road to SDN: An intellectual history of programmable networks. *ACM SIGCOMM Comput. Commun. Rev.* **2014**, *44*, 87–98. [[CrossRef](#)]
12. Said, S.B.H.; Truong, Q.H.; Boc, M. SDN-based configuration solution for IEEE 802.1 time sensitive networking (TSN). *ACM SIGBED Rev.* **2019**, *16*, 27–32. [[CrossRef](#)]
13. IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 31: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements, IEEE Standard 802.1Qcc-2018. 2018. Available online: <https://ieeexplore.ieee.org/document/8514112> (accessed on 22 October 2024).
14. Luo, F.; Wang, B.; Yang, Z. Design methodology of automotive time-sensitive network system based on OMNeT++ simulation system. *Sensors* **2022**, *22*, 4580. [[CrossRef](#)] [[PubMed](#)]
15. Wiseman, Y. Autonomous Vehicles. In *Encyclopedia of Information Science and Technology*, 5th ed.; IGI Global: Hershey, PA, USA, 2020; Volume 1, pp. 1–11.
16. Zhu, H.; Zhou, W.; Li, Z.; Li, L.; Huang, T. Requirements-Driven Automotive Electrical/Electronic Architecture: A Survey and Prospective Trends. *IEEE Access* **2021**, *9*, 100096–100112. [[CrossRef](#)]
17. Peng, Y.; Shi, B.; Jiang, T.; Tu, X.; Xu, D.; Hua, K. A Survey on In-Vehicle Time-Sensitive Networking. *IEEE Internet Things J.* **2023**, *10*, 14375–14396. [[CrossRef](#)]
18. Wang, W.; Wang, W.; Yu, H.; Wu, B.; Zou, S.; Ni, W.; Zhang, J. Joint Routing and Scheduling for In-Vehicle Networks: A Deterministic Perspective. *IEEE Trans. Intell. Veh.* **2024**, 1–15. [[CrossRef](#)]
19. Chen, J.; Zuo, Q.; Xu, Y.; Wu, Y.; Jin, W.; Xu, Y. Study of Fixed Point Message Scheduling Algorithm for In-Vehicle Ethernet. *Electronics* **2024**, *13*, 2050. [[CrossRef](#)]
20. Schweissguth, E.; Timmermann, D.; Parzyjegla, H.; Danielis, P.; Mühl, G. ILP-Based Routing and Scheduling of Multicast Realtime Traffic in Time-Sensitive Networks. In Proceedings of the 2020 IEEE 26th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), Gangneung, Republic of Korea, 19–21 August 2020; pp. 1–11. [[CrossRef](#)]
21. 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.
22. Pozo, F.; Steiner, W.; Rodriguez-Navas, G.; Hansson, H. A decomposition approach for SMT-based schedule synthesis for time-triggered networks. In Proceedings of the 2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA), Luxembourg, 8–11 September 2015; pp. 1–8.
23. Kim, H.J.; Lee, K.C.; Lee, S. A genetic algorithm based scheduling method for automotive Ethernet. In Proceedings of the IECON 2021–47th Annual Conference of the IEEE Industrial Electronics Society, Toronto, ON, Canada, 13–16 October 2021; pp. 1–5.
24. Wang, X.; Yao, H.; Mai, T.; Nie, T.; Zhu, L.; Liu, Y. Deep Reinforcement Learning aided No-wait Flow Scheduling in Time-Sensitive Networks. In Proceedings of the 2022 IEEE Wireless Communications and Networking Conference (WCNC), Austin, TX, USA, 10–13 April 2022; pp. 812–817. [[CrossRef](#)]
25. Craciunas, S.S.; Oliver, R.S.; Chmelnik, M.; Steiner, W. Scheduling real-time communication in IEEE 802.1 Qbv time sensitive networks. In Proceedings of the 24th International Conference on Real-Time Networks and Systems, Brest, France, 19–21 October 2016; pp. 183–192.
26. Schweissguth, E.; Danielis, P.; Timmermann, D.; Parzyjegla, H.; Mühl, G. ILP-based joint routing and scheduling for time-triggered networks. In Proceedings of the 25th International Conference on Real-Time Networks and Systems, Grenoble, France, 4–6 October 2017; pp. 8–17.
27. Atallah, A.A.; Hamad, G.B.; Mohamed, O.A. Routing and Scheduling of Time-Triggered Traffic in Time-Sensitive Networks. *IEEE Trans. Ind. Inform.* **2020**, *16*, 4525–4534. [[CrossRef](#)]
28. Kwon, J.-H.; Kim, H.-J.; Lee, S. Optimizing Traffic Scheduling in Autonomous Vehicle Networks Using Machine Learning Techniques and Time-Sensitive Networking. *Electronics* **2024**, *13*, 2837. [[CrossRef](#)]
29. Stüber, T.; Eppler, M.; Osswald, L.; Menth, M. Performance Comparison of Offline Scheduling Algorithms for the Time-Aware Shaper (TAS). *IEEE Trans. Ind. Inform.* **2024**, *20*, 9736–9748. [[CrossRef](#)]

30. Luo, K. Research and Implementation of Time-Sensitive Networking Configuration Management Based on IEEE 802.1Qcc. Master's Thesis, Chongqing University of Posts and Telecommunications, Chongqing, China, 2021.
31. Balasubramanian, V.; Aloqaily, M.; Reisslein, M. An SDN architecture for time sensitive industrial IoT. *Comput. Netw.* **2021**, *186*, 107739. [[CrossRef](#)]
32. Xue, J.; Shou, G.; Li, H.; Liu, Y. Enabling deterministic communications for end-to-end connectivity with software-defined time-sensitive networking. *IEEE Netw.* **2022**, *36*, 34–40. [[CrossRef](#)]
33. Hagargund, A.G.; Shet, N.S.V.; Kulkarni, M. DTPF Algorithm Based Open-Source Time-Sensitive Network Leveraging SDN Architecture. *IEEE Access* **2023**, *11*, 71037–71047. [[CrossRef](#)]
34. Dürr, F.; Nayak, N.G. No-wait packet scheduling for IEEE time-sensitive networks (TSN). In Proceedings of the 24th International Conference on Real-Time Networks and Systems, Brest, France, 19–21 October 2016; pp. 203–212.
35. IEEE Standard for Local and Metropolitan Area Networks Timing and Synchronization for Time-Sensitive Applications, IEEE Standard 802.1AS-2020. 2020. Available online: <https://ieeexplore.ieee.org/document/9121845> (accessed on 22 October 2024).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.