Advances in Computer, Signals and Systems (2026)
Clausius Scientific Press, Canada

DOI: 10.23977/acss.2026.100117
ISSN 2371-8838 Vol. 10 Num. 1

Achieving Deterministic Communication for In-Vehicle Networks with Software-Defined Time-Sensitive Networking

Chenming Jiang[1,a], Yuan Zhu[1,b], Binqi Li[1,c,*]

[1]*College of Automotive and Energy Engineering, Tongji University, No. 4800 Caoan Road, Shanghai, China*
[a]*2332996@tongji.edu.cn*, [b]*yuan.zhu@tongji.edu.cn*, [c]*bqli_tongji@tongji.edu.cn*
**Corresponding author*

Keywords: Time-Sensitive Networking, Software-Defined Networking, Deterministic Communication

Abstract: To achieve more advanced functionalities, there is a growing demand within automotive computing platforms for deterministic and reliable data transmission. Time-Sensitive Networking (TSN) is the most promising candidate to meet this demand by leveraging IEEE 802.1 Ethernet standards, which include mechanisms such as time synchronization, traffic shaping, and low-latency forwarding. This paper explores the implementation of Software-Defined Time-Sensitive Networking (SD-TSN) to achieve deterministic communication for in-vehicle networks. Typically, complex configurations involving routing and scheduling of switches are necessary to deploy TSN. SD-TSN can offer a flexible and programmable approach to network management, enabling precise control over timing constraints and quality of service (QoS) parameters. Firstly, data transmission requirements are gathered by the centralized user configuration (CUC) module to acquire traffic information. Subsequently, the centralized network configuration (CNC) module transforms the computed results of routing and scheduling into YANG model and deploys them. Finally, the automotive TSN switches can complete local configuration by parsing the received configuration messages. Through a detailed analysis and experimental validation, this study demonstrates the effectiveness of the SD-TSN architecture in enhancing deterministic communication for in-vehicle networks.

1. Introduction

With the continuous advancement of automotive intelligence and connectivity, the demand for high-speed and reliable data transmission within vehicles has increased significantly. Various applications, such as advanced driver assistance systems and infotainment systems, now necessitate more robust communication networks to ensure optimal performance [1]. However, traditional automotive bus technologies have progressively demonstrated limitations such as restricted bandwidth and inefficiencies in data transmission. In this regard, Time-Sensitive Networking (TSN) has emerged as a focal area of research, aiming to enhance the real-time data transmission

capabilities of Ethernet technology. And TSN is positioned to become the backbone network used in the next-generation automotive electronic and electrical architecture [2].

Time-Sensitive Networking encompasses various mechanisms aimed at ensuring deterministic real-time communication over Ethernet. Key mechanisms include time synchronization protocols such as IEEE 802.1AS, which achieves network-wide clock synchronization with sub-microsecond accuracy in distributed systems. Additionally, IEEE 802.1Qbv standard provides a time-aware shaper (TAS), enabling strict prioritization and bandwidth allocation to critical traffic flows. As shown in Figure 1, this mechanism categorizes traffic into different queues according to their priority, the transmission of each queue is controlled by the gate control list (GCL) within predefined periodic cycles. TAS ensures that high-priority traffic, essential for applications requiring stringent latency guarantees (such as industrial control systems or real-time multimedia), is granted access to network resources at predetermined time slots. Thus, a well-designed GCL can effectively facilitate the deployment of converged real-time and non-real-time traffic on a single Ethernet network infrastructure.

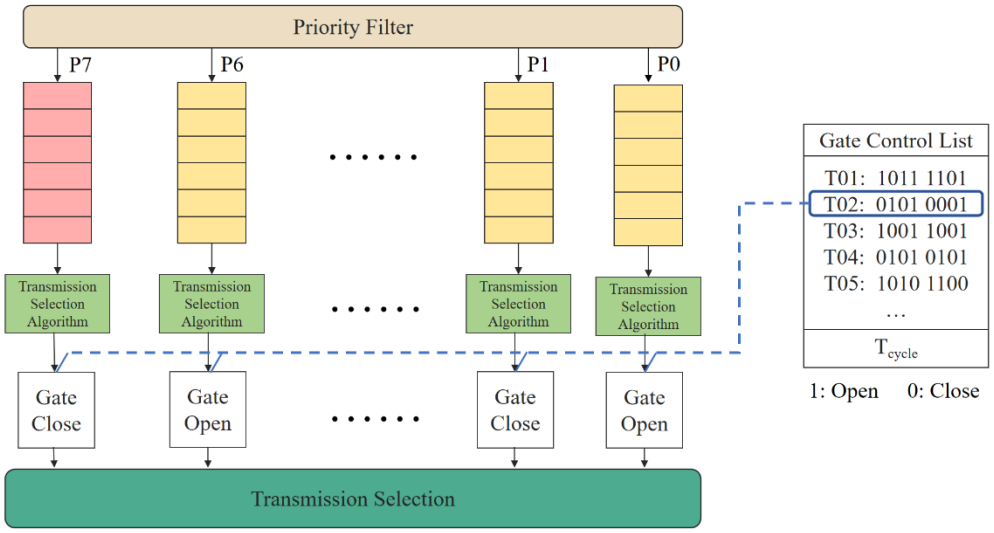

Figure 1: Time-aware shaper mechanism.

Scholars have conducted extensive research on the scheduling and resolution of TAS mechanism to enable the deterministic end-to-end latency. Craciunas et al. meticulously analyzed the transmission of critical flows in IEEE 802.1Qbv and derived a constraint model for computing offline scheduling schemes [3]. This constraint model has become the mainstream method for deterministic resolution of transmission offsets for time-sensitive flows. The study also proposed frame isolation and flow isolation as protective measures to mitigate TAS vulnerabilities. Schweissguth et al. [4] introduced a novel Integer Linear Programming (ILP) formulation to concurrently solve the routing and scheduling of time-sensitive data flows. Compared to separate routing and scheduling approaches, this joint solving method offers greater solution space and addresses problems that are unsolvable under conventional shortest-path routing methods.

Once satisfactory solutions are obtained through routing and scheduling algorithms, they must be configured into TSN supported switches within the network. With the increasing proliferation and complexity of real-time data transmission applications, the configuration demands for TSN are becoming increasingly stringent. Traditional network configuration implemented manually by network administrators are often constrained by the inherent complexities of hardware devices, which pose challenges in meeting the exacting requirements of real-time data transmission. As an innovative network management paradigm, Software-Defined Networking (SDN) offers a fresh

approach to tackling this challenge through the separation of the control plane and the data plane. To achieve configuration of Time-Sensitive Networking (TSN) based on the principles of SDN, the IEEE 802.1Qcc standard [5] proposes three configuration models: distributed, centralized, and fully centralized. The centralized configuration model uses a centralized user configuration (CUC) module to gather requirements from end devices and then passes this information to the centralized network configuration (CNC) module. The CNC uses remote network management protocols to configure the switches.

Based on the IEEE 802.1Qcc standard, Luo Kun [6] designed a TSN configuration management system that implements topology discovery, YANG model configuration, and traffic scheduling functions. This system is relatively complex and demands high requirements on both endpoints and switches, making it difficult to deploy on resource-constrained platforms like automotive embedded systems. To meet the reconfigurability needs of the Industrial Internet of Things, Venkatraman et al. integrated routing and scheduling algorithms within the SDN controller to calculate TSN configurations in real time [7]. Junli et al. validated through simulations that SDN-based TSN configurations can meet the end-to-end latency requirements of time-sensitive data streams [8].

In fact, the majority of current research on TSN remains confined to the simulation level, lacking validation on physical platforms. This paper proposes a software-defined TSN (SD-TSN) architecture to achieve deterministic transmission for in-vehicle networks. Additionally, an ILP-based scheduling algorithm is introduced that considers the outside influence in real-world scenarios. This algorithm aims to reduce the vulnerability of TAS in practical applications and we integrate it into the proposed architecture for calculating transmission offsets of time-sensitive data flows on the forwarding paths.

The remainder of this paper is organized as follows: Section 2 formulates the scheduling model used to compute the forwarding paths and transmission offsets. Section 3 elaborates the working process of the SD-TSN architecture. Section 4 evaluates the performance of TAS on deterministic transmission based on a physical platform. Finally, we make a conclusion and a vision of the future work in Section 5.

2. Problem Formulization of TAS Scheduling

2.1. System Model

With the increasing functionality of automotive electronic systems, the data transmission requirements have also been continuously rising. Traditional distributed architectures can no longer meet the communication needs of the next-generation vehicles. Hence, the zonal architecture concept has been proposed [9]. It utilizes Ethernet, which offers higher bandwidth, as the backbone for data transmission, facilitating the deployment of high-performance platforms. In this paper, an automotive zonal architecture shown in the Figure 2 is selected as the network topology. This topology is composed of endpoints, switches, gateway, and PC. To describe the network topology information, a directed graph model can be utilized to store its structure along with the attributes of nodes and links within it.

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

where G represents the entire topology model; V denotes the set of all nodes within the topology; E signifies the collection of all links in the topology. Each node 12 within the topology model is

required to retain its respective name, IP address, and MAC address information, whereas each link e must retain details pertaining to its source node, destination node, source port, destination port, and bandwidth.

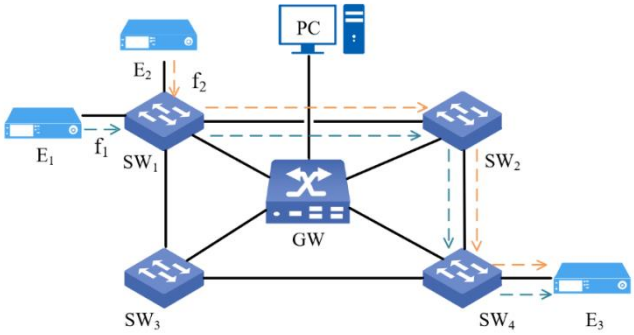

Figure 2: An automotive zonal architecture.

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

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

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

2.2. Formulization of Scheduling Model

Scheduling for time-sensitive data flows presents a classical job shop problem (JSP) that is NP-complete and is recognized as one of the most challenging combinatorial optimization problems [10]. Integer linear programming (ILP) is widely acknowledged as an effective approach for tackling this issue. To ensure the precise temporal performance of deterministic communication, the constraints necessary for constructing the ILP-based scheduling model are formulated in this subsection. To reduce computational complexity, this study segregates routing and scheduling computations, employing elementary shortest-path algorithms for determining forwarding paths of time-sensitive data flows.

2.2.1 Transmission Start Constraint

The start transmission time on each link plus the transmission delay has to fit within the flow period to ensure that there is enough time left for transmission.

$$\begin{gathered} \forall f_k \in F, \ \forall e_m \in R_k : \\ 0 \leqslant t_k^m \leqslant f_k.T - \frac{f_k.load}{e_m.bd}, \end{gathered} \tag{3}$$

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

2.2.2 Flow Isolation Constraint

In practice, a data flow may consist of one or multiple frames. Instead of employing frame isolation constraints which offers higher fault tolerance, this paper opts for flow isolation

constraints to reduce the number of decision variables and thereby simplify the scheduling model. Specifically, at any given time, a switch output port's buffer queue is restricted to storing 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.

$$\begin{gathered}\forall f_k, f_l \in F \mid k \neq l, \\ \forall e_m \in (R_k \cap R_l) \mid e_m.v_s \neq (f_k.src \cup f_l.src), \\ \forall \mu \in \left[0, \frac{hp_{kl}}{f_k.T}\right), \forall \nu \in \left[0, \frac{hp_{kl}}{f_l.T}\right): \\ \left(t_k^m + \mu^* f_k.T + \frac{f_k.load}{e_m.bd} \leqslant t_l^q + \nu^* f_l.T\right) \vee \\ \left(t_l^m + \nu^* f_l.T + \frac{f_l.load}{e_m.bd} \leqslant t_k^p + \mu^* f_k.T\right),\end{gathered} \tag{4}$$

where hp_{kl} represents the hyper-period between flows f_k and f_l, which equals the least common multiple of their respective periods. Variables μ and ν denote the instances of flows f_k and f_l within this hyper-period. And the index p and q signify the respective previous-hop links for flows f_k and f_l before arriving at link e_m.

2.2.3 Link Resource Constraint

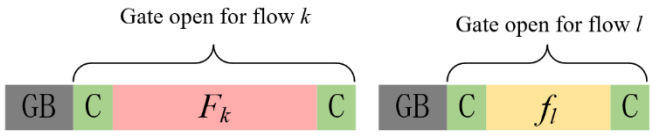

Figure 3: Guard band and compensation in GCL.

To ensure conflict-free transmission, the time slots assigned to all flows routed through the same link must not overlap temporally. 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.

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

where, GB represents the guard band, defined as the duration required to transmit a maximum 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 3, 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 TAS mechanism when applied in practical scenarios.

2.2.4. 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 meters, the propagation time through each link is negligible at the nanosecond scale, hence not considered in this study.

$$\forall f_k \in F,\ \forall e_m, e_n \in R_k \mid e_n.v_s == e_m.v_d :$$
$$t_k^m + \frac{f_k.load}{e_m.bd} + d_{proc} \leqslant t_k^n, \tag{6}$$

where d_{proc} represents the processing delay within 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.

2.2.5 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 end-to-end latency requirements are met for time-sensitive data flows.

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

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

3. Software-Defined TSN Architecture

To ensure the real-time transmission of critical data streams in vehicular time-sensitive networking, intricate configurations involving the routing and scheduling of switches are indispensable. Given the shortcomings of static configuration methods characterized by high workload and low flexibility, we propose a centralized configuration architecture shown in Figure 4 grounded in SDN principles. This architecture aims to dynamically control the forwarding path and reserve bandwidth in-vehicle time-sensitive networks to fulfill stringent deterministic transmission demands of critical data streams.

3.1. Working Principle of the SD-TSN Mechanism

As depicted in Figure 4, the SD-TSN mechanism operates on vehicular networks by coordinating interactions among four entities: end nodes, switches, gateways, and controllers, facilitating the execution of Time-Aware Shaper (TAS). The detailed operational principles are elaborated as follows.

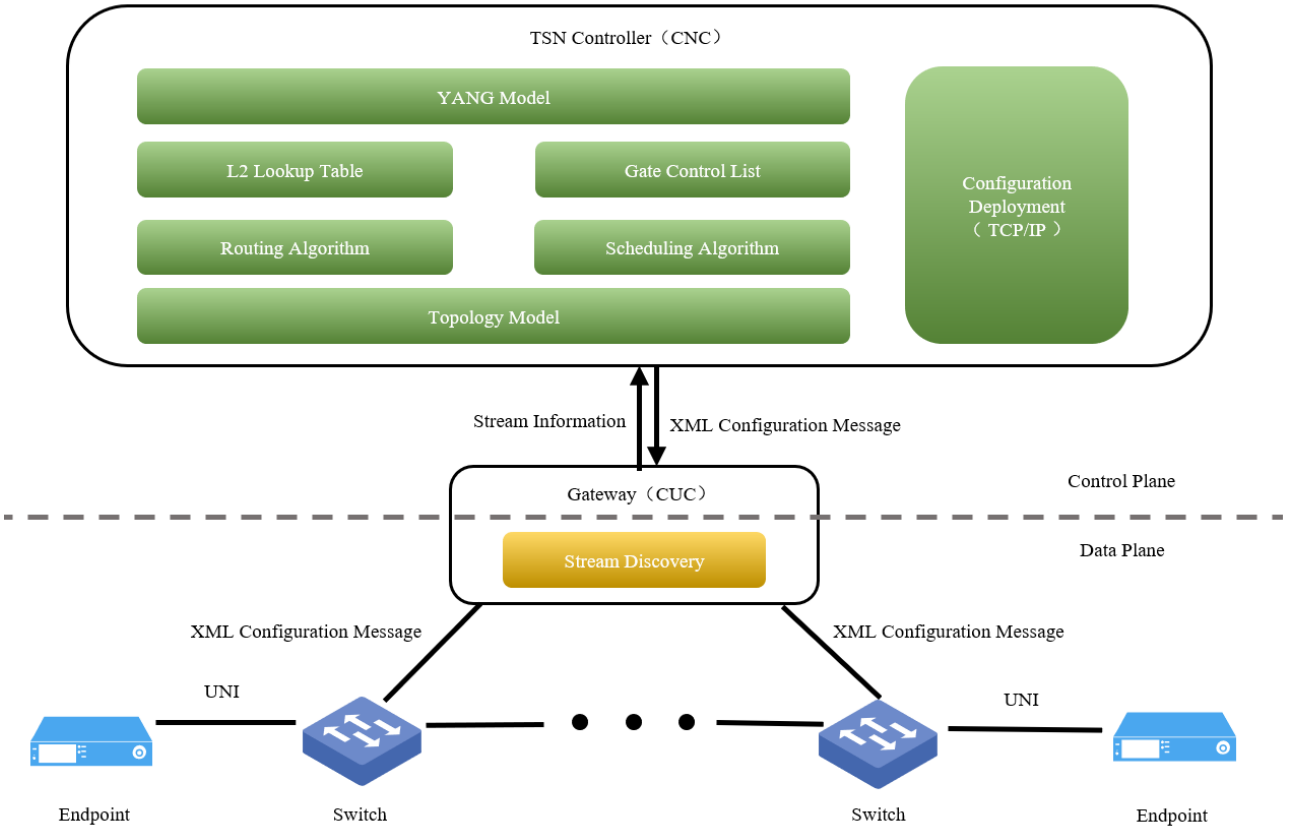

Figure 4: Software-defined TSN architecture for in-vehicle network.

1) Achieving precise clock synchronization across the entire network is paramount. In time-sensitive networks, a unified clock reference forms the cornerstone 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 its own transmission requirements.

3) The gateway incorporates the CUC module responsible for the stream discovery process, by establishing connections to switches for real-time traffic information gathering. 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 based on the calculation results and populates them into YANG models to generate configuration for switches. These 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 configuration of the local L2 lookup table and gate control list within TAS mechanism.

3.2. Stream Discovery

In the SDN architecture, the CUC utilizes the UNI for network-wide stream discovery and transmits gathered traffic information to the CNC. To enhance traffic information collection, the proposed centralized configuration model situates the CUC within the in-vehicle gateway. Given

that the gateway directly interfaces with all switches in the network, Talkers and Listeners can engage with the CUC via these switches to exchange traffic details. The protocol employed for user-CUC interaction varies based on upper-layer applications operating on the user's system. For example, if communication middleware like SOME/IP or DDS is utilized, the middleware's discovery mechanism may be leveraged. Due to space constraints, this aspect will not be extensively discussed in this paper.

3.3. Path Control and L2 Lookup Table Configuration

Path control refers to the explicit routing of data streams through a network using specially designed forwarding tables, rather than relying on the self-learning capabilities of switches. Additionally, bandwidth reservation requires predetermined routing paths to allocate the corresponding bandwidth resources for data streams. To achieve path control, the forwarding tables on all switches for each time-sensitive data stream should be configured in advance. Therefore, this paper proposes a method for dynamically configuring switch L2 lookup tables based on the SD-TSN architecture. In this architecture, the TSN controller can dynamically adjust the L2 lookup tables of switches based on network traffic and topology information, avoiding the complexity of configuring each switch individually and effectively simplifying the network management process.

Regarding routing algorithms, extensive research has been conducted in academia, proposing optimization approaches such as shortest path algorithms, load-balancing-aware routing algorithms, and multipath routing algorithms from different perspectives. Detailed discussion on route calculation is beyond the scope of this paper; readers are encouraged to refer to relevant literature to select suitable routing algorithms. This paper defines the YANG model shown in Table 1 for configuring L2 lookup table entries on TSN switches, specifying mandatory attributes such as index number, destination MAC address, source port, VLAN ID, and destination port information.

Table 1: L2 lookup table YANG model.

Entry	Node 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 srource port through which this stream enters the switch.
vlanid	leaf	Uint16	the VLAN id used by this stream.
destport	leaf	Uint16	the port through which this stream departures the switch.

3.4. Bandwidth Reservation and GCL Configuration

Bandwidth reservation technology refers to the reasonable allocation and scheduling of network bandwidth resources by reserving a certain amount of bandwidth for real-time data streams, ensuring their timely transmission. The bandwidth reservation mechanism based on traffic scheduling involves scheduling and managing network traffic through traffic scheduling algorithms to achieve bandwidth reservation and allocation. Scholars have extensively researched and optimized scheduling mechanisms such as Qav, Qbv, and Qch in TSN. This paper focuses on the Time-Aware Shaper (TAS) proposed in the Qbv protocol, deploying the ILP scheduling algorithm in section 2 on TSN controller for computing Gate Control Lists (GCL). By dynamically calculating and configuring switch GCLs based on traffic and topology information, the aim is to dynamically adjust bandwidth allocation to ensure real-time transmission of time-sensitive data streams.

To achieve bandwidth reservation based on time-aware shaping, Table 2 defines a YANG model for GCL configuration. The YANG model specifies initially which port on the switch to configure. Subsequently, a list format stores the TAS scheduling entries, wherein each entry includes its index,

the status of queue gate controls, and the duration of the gate status will be maintained.

Table 2: Gate control lists YANG model.

Entry	Node type	Value type	Description
port	leaf	Uint16	the port number on the switch.
tasScheduleEntry (list)	index	leaf	Uint16
	triggerTime	leaf	Uint32
	gateStatus	leaf	Uint8

Upon configuration completion, 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 configuration of gate control lists based on these timings, we introduce the GCL generation algorithm depicted as below:

```
Input: G, F, T
Output: O, YANG
Begin
for f in F do
    hp = Lcm(hp, f.T);
end for
for f in F do
    n = hp / f.T;
    for each T[f][e] in T[f] do
        for i = 0 to n do
            O[e].insert(T[f][e] + i * f.T);
        end for
    end for
end for
for each e in E do
    YANG.port = LookupPort(e.vs, vd);
    accuLen = 0;
    i = 0;
    for each t in O[e] do
        GB = MaxFrame / e.bd;
        duration = f.load / e.bd + 2 * C;
        YANG[i].index = i++;
        YANG[i].tt = t - accuLen - GB – C;
        YANG[i].gs = 01111111   // non-TS traffic;
        YANG[i].index = i++;
        YANG[i].tt = GB;
        YANG[i].gs = 00000000;   // guard band
        YANG[i].index = i++;
        YANG[i].tt = duration;
        YANG[i].gs = 10000000;   // TS traffic
        accuLen = t + duration + C;
    end for
end for
return YANG
```

This algorithm takes the network topology G, flows set F, and scheduling result T as inputs,

and outputs YANG model information for GCL configurations. Current mainstream scheduling algorithms target periodic time-sensitive (TS) data streams for scheduling. To simplify the scheduling model complexity, the output result T only contains the transmission offset of flow f on path e within the first cycle. However, all GCL's cycle on switches throughout the network must remain consistent and equal to the least common multiple of all periodic time-sensitive flow cycles, termed the hyper-period. Algorithm calculates this hyper-period hp in lines 1-3. Then, lines 4-11 compute the time points O when gates with priority 7 are open for all TS-traffic across the entire hyper-period. With these time points determined for each path, GCL configurations are generated based on the points in $O[e]$. Considering varying bandwidth capacities across links in in-vehicle networks, the algorithm computes the required durations for gate openings and guard band for each link. Since compensation has already been accounted for within the link resource constraint, the compensation values can be included in the calculation of duration at 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; lines 25-27 add entries for time-sensitive traffic.

Upon completion of YANG generation for GCL configurations, it is converted into XML format and deployed to TSN switches via TCP connection, thereby achieving bandwidth reservation based on GCL.

4. Experimental Evaluation

4.1. Experimental Platform

To validate the efficacy of the proposed software-defined time-sensitive networking architecture, we established a physical validation platform for automotive Ethernet illustrated in Figure 5. This platform features essential components including two Raspberry Pi 5 boards, four NXP SJA1110 switches, two NXP S32G2 evaluation boards, and a PC (linked to the central S32G2 via the yellow Ethernet cable). The NXP SJA1110 is an automotive Ethernet switch that supports some basic TSN protocols like IEEE 802.1AS and IEEE 802.1Qbv. On the other hand, S32G2 is a specialized automotive processor chip tailored by NXP for forthcoming in-vehicle computing and networking architectures. Its robust computational capabilities render it suitable for deployment as an in-vehicle gateway.

This experimental platform is constructed in accordance with the zonal architecture topology delineated in Figure 2. In this configuration, the SJA1110 devices function as TSN switches, two Raspberry Pi 5 boards operates as endpoint E1 and E3, the leftmost S32G2 serves as endpoint E2, and the central S32G2 acts as the gateway interfacing with the PC that hosts the TSN controller. The bandwidth of all Ethernet links in this platform is 100 Mbps.

In order to simulate the whole working process of the SD-TSN mechanism, we chose to deploy data distribution services (DDS) at both endpoints and gateways to serve as data producers and consumers in application layer. What's more, this deployment enables the gateway's CUC to utilize DDS discovery messages for stream discovery functionality. A TSN controller is deployed on the PC to act the CNC module, executing routing and scheduling algorithms along with the configuration algorithm proposed in this paper. The network topology is modeled by the igraph which is a free and open source package for network analysis. And the scheduling model is built and solved by Gurobi Optimizer which is a professional solver for ILP optimization problem. To deploy the configurations, the PC connects to the gateway via Ethernet and transmits configuration messages utilizing a TCP/IP connection. Due to the lack of support for the NETCONF network management protocol in SJA1110, we developed parsing capabilities for YANG models in its software stack. Subsequently, it can dynamically invoked interfaces from its SDK to configure the

L2 lookup table and TAS to transmit the time-sensitive data.

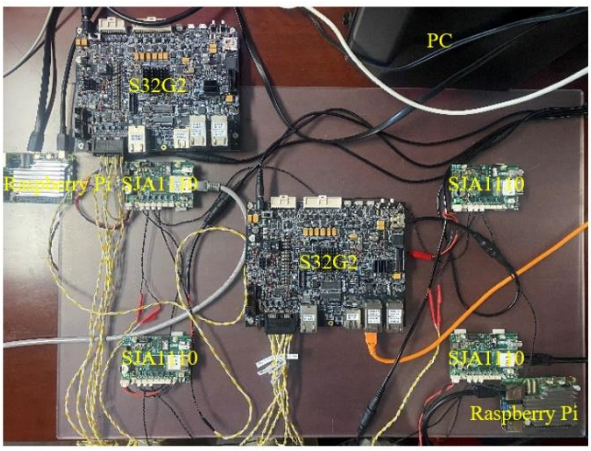

Figure 5: Validation Platform for Automotive Ethernet.

4.2. Experimental Implementation

To validate the efficacy of the SD-TSN mechanism proposed in this study for achieving deterministic transmission of time-sensitive data streams, as depicted in Figure 2, we deployed two data flows in the network, where flow 1 represents the time-sensitive flow and flow 2 represents the non-time-sensitive interference flow. We deploy FastDDS in the application layer to construct these two flows, and the attributes of these two flows are configured according to Table 3.

Table 3: Experimental flows attributions.

Name	Talker	Listener	Priority	Period/ms	Payload/Byte
Flow 1	E1	E3	7	50	1024
Flow 2	E2	E3	0	10	3200~102400

Flow 1 represents a time-sensitive data stream of highest priority, while Stream 2 serves as an interfering stream with lower priority and variable payload size to simulate the state of the network from idle to busy. In the experiments, we set the maximum allowable latency for flow 1 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 as E2 - SW1 -SW2- SW4- E3.

4.3. Experimental Results

In this experiment, 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 two different scheduling mechanisms: the traditional Strict Priority (SR) where TSN is not required and the time-aware shaper (TAS) which is facilitated by the SD-TSN architecture proposed in this paper. We will compare the latency and jitter of time-sensitive streams under varying degrees of interference to assess their impact on deterministic transmission.

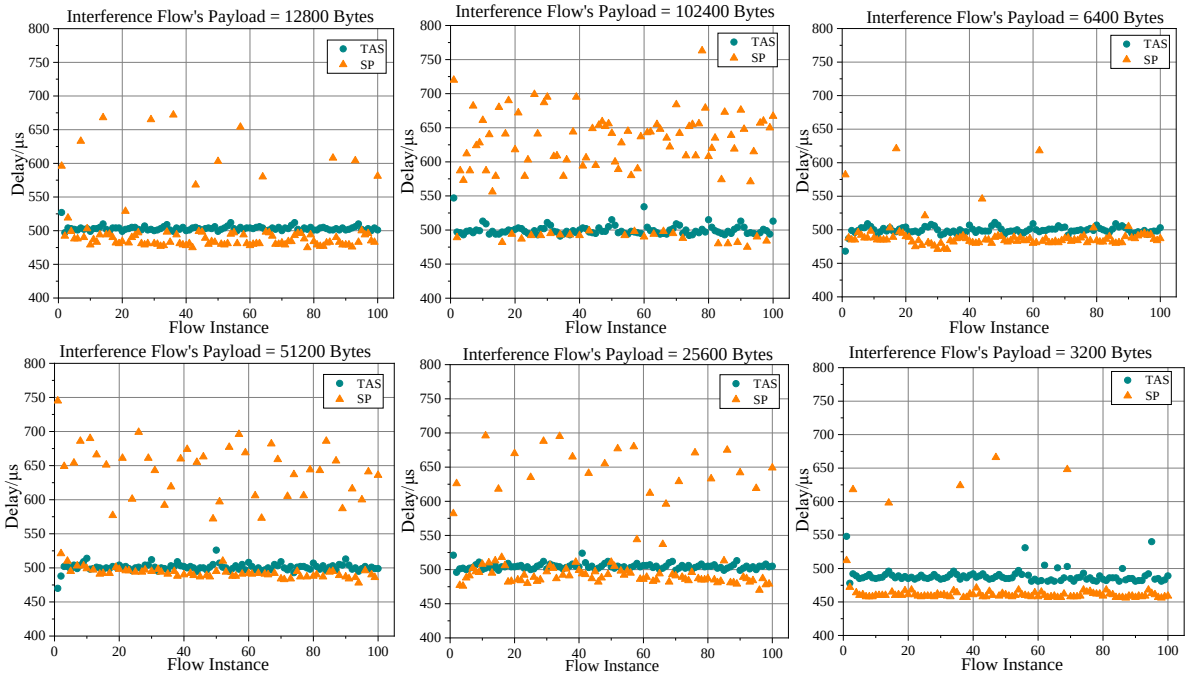

Figure 6: The end-to-end latency of TT flow instances.

Figure 6 depicts the latency test results of time-sensitive data flow as the load of interfering traffic increases, transitioning from link states of idle to full. When the load of interfering traffic is low, both SP and TAS mechanisms demonstrate relatively small and stable end-to-end delays. However, as the interfering traffic load increases beyond 12,800 bytes, the end-to-end latency under strict priority scheduling deteriorates significantly, exhibiting not only increased values but also intensified jitter. This degradation occurs because while the TT flow is prioritized for transmission, it must wait for interfering traffic frames to complete transmission before it can begin its own transmission. This introduces additional queuing delays for the TT flow, which are unpredictable, thereby inevitably contributing to jitter in the end-to-end latency.

In comparison with strict priority (SP), it is evident that after adopting the SD-TSN mechanism, TAS effectively 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 scheduling model for TAS, which allocates dedicated transmission slots for the TT flow and ensures its end-to-end latency requirements are met through constrained scheduling.

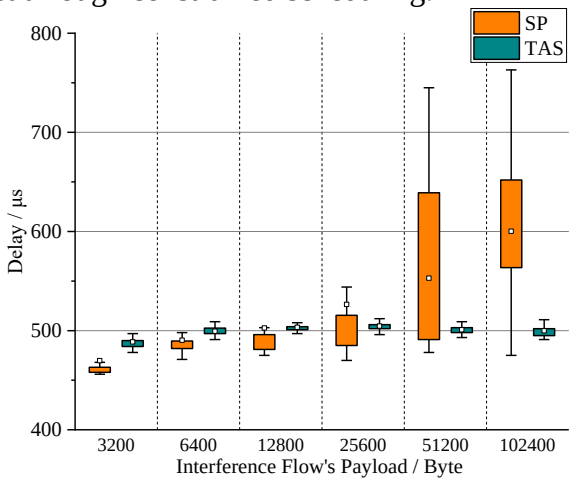

Figure 7: Latency distribution across different interference traffic loads

To further demonstrate the effectiveness of our study in achieving deterministic transmission, as illustrated in Figure 7, we plotted the distribution of latency under different levels of interference traffic using boxplots. It is evident that with increasing interference intensity, 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 flow stabilizes around 500 microseconds with minimal fluctuation, indicating compliance with the requirements for deterministic transmission.

5. Conclusions

Time-sensitive Networking (TSN) is poised to become the backbone network in the next generation of automotive E/E architecture. To address the complexities of its configuration in practical applications, this paper proposes a SD-TSN architecture to dynamically configure in-vehicle TSN networks. To achieve deterministic transmission for time-sensitive data flows, we formulate an ILP-based scheduling model for the TAS mechanism. What's more, the YANG models for configuring L2 lookup tables and gate control list are established to implement the path control and bandwidth reservation. The experiments conducted on a vehicular physical platform demonstrate that the SD-TSN mechanism can ensure that time-sensitive flows are not interfered by other traffic, effectively enhancing the determinism of data transmission in vehicular networks.

However, the practical application of TSN in automotive settings still has a long way to go. One crucial aspect is that, the performance of TSN is influenced by various factors, necessitating further optimizations in tasks scheduling within the operating system and traffic shaping algorithms to facilitate the deployment of time-sensitive networks in automotive applications. We incorporate compensation values into the scheduling algorithm to mitigate the influence of non-ideal factors. Future work will explore the relationship between compensation values and other factors, aiming to optimize their selection through the establishment of mathematical models.

Acknowledgements

This work was funded by the Perspective Study Funding of Nanchang Automotive Institute of Intelligence and New Energy, Tongji University (TPDTC202211-07).

References

[1] Zhang, Chengmin, et al. "Deterministic communications for in-vehicle network: Overview and challenges." 2021 2nd International Conference on Artificial Intelligence and Information Systems. 2021.

[2] Zhu, Hailong, et al. "Requirements-driven automotive electrical/electronic architecture: a survey and prospective trends." IEEE Access 9 (2021): 100096-100112.

[3] Craciunas, Silviu S., et al. "Scheduling real-time communication in IEEE 802.1 Qbv time sensitive networks." Proceedings of the 24th International Conference on Real-Time Networks and Systems. 2016.

[4] Schweissguth, Eike, et al. "ILP-based joint routing and scheduling for time-triggered networks." Proceedings of the 25th International Conference on Real-Time Networks and Systems. 2017.

[5] "IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks – Amendment 31: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements," in IEEE Std 802.1Qcc-2018 (Amendment to IEEE Std 802.1Q-2018 as amended by IEEE Std 802.1Qcp-2018). pp.1-208, 31 Oct. 2018.

[6] Luo Kun, Research and Implementation of Time-Sensitive Networking Configuration Management Based on IEEE 802.1Qcc[D]. Chinese Master's Theses Full-text Database, 2021.

[7] Balasubramanian, Venkatraman, Moayad Aloqaily, and Martin Reisslein. "An SDN architecture for time sensitive industrial IoT." Computer Networks 186 (2021): 107739.

[8] Xue, Junli, et al. "Enabling deterministic communications for end-to-end connectivity with software-defined time-sensitive networking." IEEE Network 36.2 (2022): 34-40.

[9] M. Bornemann, Zone Controllers Build Bridge to Tomorrow's Technology, 2021.

https://www.aptiv.com/docs/default-source/white-papers/2021_aptiv_whitepaper_zonecontroller.pdf.
[10] W. Brinkkötter, P. Brucker, Solving open benchmark instances for the job-shop problem by parallel head–tail adjustments, Journal of Scheduling 4 (2001) 53–64.

Definitions/Abbreviations

TSN	Time-Sensitive Networking
GCL	Gate Control Lists
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
SRP	Stream Reservation Protocol
ILP	Integer Linear Programming
TT	Time Sensitive
TAS	Time-Aware Shaper
SP	Strict Priority
DDS	Data Distribution Service
SOME/IP	Service-Oriented Middleware over IP