World Electric Vehicle Journal

MDPI

Article

Time-Sensitive Network Simulation for In-Vehicle Ethernet Using SARSA Algorithm

Chen Huang [1,2], Yiqi Wang [2,*] and Yuxin Zhang [1]

[1] College of Automotive Engineering, Jilin University, Changchun 130012, China; huangchen@ujs.edu.cn (C.H.); yuxinzhang@jlu.edu.cn (Y.Z.)
[2] Institute of Automotive Engineering, Jiangsu University, Zhenjiang 212013, China
* Correspondence: wangyiqi23@163.com

Abstract: In order to more accurately analyze the problem of time delay simulation and calculation in the time-sensitive network (TSN) of vehicular Ethernet, a TSN reservation class data delay analysis model improved based on the State–Action–Reward–State–Action (SARSA) reinforcement learning algorithm is proposed. Firstly, the TSN data queue forwarding delay model and reservation class data delay analysis intelligent body model are established, then the TSN traffic scheduling mechanism is improved by the SARSA reinforcement learning algorithm, and the improved TSN network reservation class data analysis model is established for the uncertainty of traffic scheduling in the network; finally, the fitting performance of the proposed method is verified by simulation and experimental validation. The results show that the deviation between the two is less than 5% under different BE loads, i.e., the established reservation class data delay analysis model is able to correctly fit the scheduling mechanism of the vehicle-mounted TSN network, which proves the reasonableness of the model simulation.

Keywords: in-vehicle Ethernet; time-sensitive networks; SARSA algorithm; simulation computation; intelligent body modeling

check for updates

Citation: Huang, C.; Wang, Y.; Zhang, Y. Time-Sensitive Network Simulation for In-Vehicle Ethernet Using SARSA Algorithm. *World Electr. Veh. J.* **2024**, *15*, 21. https://doi.org/10.3390/wevj15010021

Academic Editor: Joeri Van Mierlo

Received: 11 December 2023
Revised: 29 December 2023
Accepted: 2 January 2024
Published: 8 January 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/).

1. Introduction

In recent years, there has been a rapid development of intelligent and connected vehicles, leading to a new trend in the automotive engineering field. An increasing number of electronic control systems, such as antilock braking systems (ABSs), advanced driver assistance systems (ADASs), and in-vehicle infotainment systems (IVIs) [1], have been implemented in automobiles. Although these electronic control systems enhance the handling, safety, and comfort of vehicles, they also result in the installation of a greater number of electronic control units (ECUs) in automobiles [2]. Therefore, smart connected vehicles require high bandwidth from the in-vehicle network, while also necessitating a network that is reliable enough to maintain low latency and jitter [3]. The traditional in-vehicle network has gradually become inadequate in meeting the diverse needs of products, thus emphasizing the need to expand and standardize the protocols for in-vehicle networks based on Ethernet. With these requirements in mind, it is necessary to conduct research and apply advances in in-vehicle Ethernet.

For in-vehicle Ethernet, the IEEE802.1 Working Group has been working on the development of Ethernet Audio Video Bridging (AVB) technology [4–11]. AVB technology supports various features such as clock synchronization, bandwidth reservation, low-latency traffic specification, and more. The AVB protocol cluster has been extended with the introduction of TSN technology, which has attracted the interest of many researchers. TSN builds upon the foundation of AVB technology and incorporates new mechanisms and enhancements in IEEE 802.1 bridges and end stations. This ensures a low-latency, high-reliability, bandwidth-isolated, and zero-blocking-loss network, with deterministic

transmission time for Ethernet Local Area Networks (LANs). TSNs offer an enhanced functional implementation of traditional Ethernet in specific application scenarios such as automotive. Additionally, TSNs enable a single physical network for applications with different communication requirements, thus providing multiple network solutions [12]. Therefore, with its distinct benefits, TSN is increasingly utilized in various fields such as autonomous driving, intelligent cockpit, vehicle-to-circuit coordination, and more [13–16].

Regarding TSN technology, there have been numerous studies conducted both domestically and internationally. Scholars have focused on researching and discussing TSN's practical application scenarios and trends [17]. Regarding delay analysis, Jahanzaib Imtiaz examined the delay of AVB in industrial real-time communication using the Priority Queuing (PQ) and Class Based Queuing (CBQ) methods [18]. Dorin Maxim studied issues related to upper bounds on the delay in AVB data streams and the transmission delay of different types of CDT time slots [19]. Sivakumar Thangamuthu conducted an analysis and comparison of the worst-case end-to-end delay and the impact on AVB traffic in TSN networks for three shapers: Burst Limiting Shaper (BLS), Time Aware Shaper (TAS), and Peristaltic Shaper (PS) [20]. Daniel Thiele thoroughly examined the worst-case delay in TSN networks using the TAS and PS mechanisms. This analysis encompasses the impact of various service types as well as the impact within the same services [21]. Luxi Zhao and Paul Pop proposed a network calculus-based approach to calculate the worst delay value for AVB data streams in TSN networks. They conducted an analysis and comparison of the upper bounds of delay in two different cases: with and without the frame preemption mechanism [22]. Ehsan Mohammadpour analyzed the worst end-to-end delay for reservation class traffic in TSN networks, focusing on the CBS and asynchronous traffic shaping mechanisms [23]. Yuefei Wang proposed a hierarchical scheduling mechanism and performed an analysis of the worst end-to-end delay for reservation class traffic. This analysis takes into account the influence of FIFO (First In First Out), CBS (Credit-Based Shaper), and TAS mechanisms [24]. Hao-Liang Xu put forward a frame preemption delay characterization model utilizing a hybrid probability model. Additionally, the impact of delay influencing factors on the component weights of the hybrid probability model is analyzed through simulation [25].

TSN guarantees real-time control of data traffic based on AVB, but it can impact the transmission delay of reservation class data within the network. Currently, there is ample literature on the time delay of reservation class traffic in TSNs, but much of it focuses on extreme cases, considering all possible factors that may disrupt reservation class traffic. In vehicular TSN networks, on the other hand, due to their unpredictability, it is difficult to accurately model or analyze the delay of reserved class data under extreme conditions. Therefore, it is essential to account for traffic scheduling randomness, develop a suitable set of delay analysis models, and conduct further research on the transmission delay of reservation class data.

This paper aims to analyze the data flow scheduling mechanism and algorithms within TSN networks. Due to non-deterministic scheduling challenges, this paper proposes an intelligent SARSA reinforcement learning algorithm for delay analysis of reservation class data in vehicular TSN networks, which will provide a more accurate simulation and analysis of TSN network.

2. TSN Data Queue Forwarding Latency Modeling

2.1. Analysis of Mathematical Models

The *TAS* scheduling mechanism guarantees the real-time transmission of Control Data Traffic (CDT) while non-CDT traffic is subject to the First-In-First-Out (*FIFO*), Credit-Based Shaper (*CBS*), and Time Aware Shaper (*TAS*) mechanisms, which can contribute to significant latency. Consequently, the primary factors influencing queue delays in non-CDT traffic include *FIFO* delay, *CBS* delay, *TAS* delay, and data transmission delay [26].

$$R = R_{FIFO} + R_{CBS} + R_{TAS} + R_T \tag{1}$$

where R is the queue forwarding delay for data frames, R_{FIFO} is the $FIFO$ delay, R_{CBS} is the CBS delay, R_{TAS} is the TAS delay, and R_T is the data frame forwarding delay.

2.1.1. Mathematical Modeling of *FIFO* Delay

The *FIFO* delay formula [27] in equation is shown below:

$$R_{FIFO} = \frac{M_j}{C} \tag{2}$$

where C is the transmission rate and M_j is the frame length of the same type of data interference frame.

Taking into account the impact of the *CBS* algorithm [28] on the Stream Reservation (SR) class data flow, the aforementioned equation requires correction. As illustrated in Figure 1, when multiple SR class frames with the same priority enter the forwarding queue, the second SR class frame must wait for the credit value of the first SR class frame to reach a non-negative value from a negative value after completion of transmission, before it can initiate transmission. The recovery time is determined by the ratio of the absolute value of the minimum credit value to the credit value growth rate. Considering the *CBS* algorithm, the modified *FIFO* delay is as follows:

$$R_{FIFO} = \frac{M_j}{C} + \left| \frac{loCredit}{idleSlope} \right| = \frac{M_j}{C}\left(1 + \frac{sendSlope}{idleSlope}\right) \tag{3}$$

where $loCredit$ is the rate of credit accumulation, $idleSlope$ is the rate of credit accumulation, and $sendSlope$ is the rate of credit consumption.

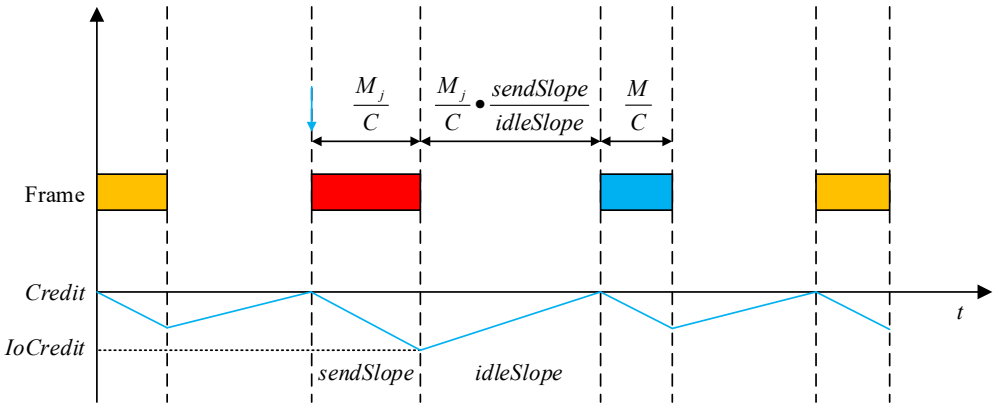


Figure 1. Process of forwarding data frames by FIFO mechanism considering *CBS* mechanism.

2.1.2. Mathematical Model of *CBS* Delay

As depicted in Figure 2, the *CBS* algorithm classifies data streams into three categories: L (low) priority, M (medium) priority, and H (high) priority, which correspond to BE, SR_B, and SR_A, respectively. During the transmission of a BE frame, both a Class B frame (B_1) and a Class A frame (A_1) are simultaneously added to the queue [29]. Therefore, the *CBS* delay can be calculated as follows:

$$R_{CBS} = t_{LPB} + t_{HPB} \tag{4}$$

where t_{LPB} is the L-level data stream delay, and t_{HPB} is the H-level data stream delay.

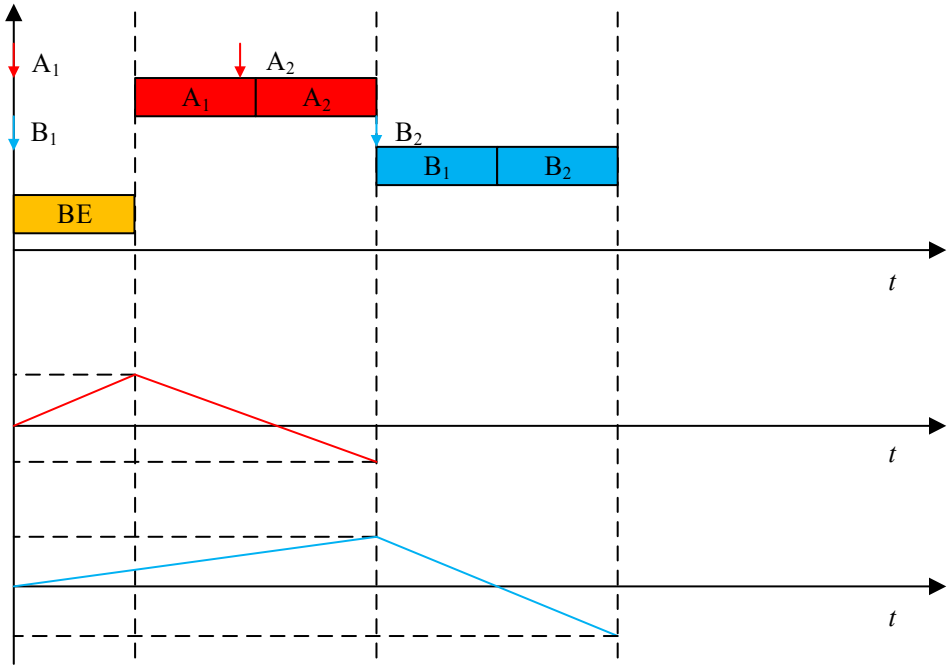


Figure 2. Process of forwarding data frames by *CBS* mechanism.

2.1.3. Mathematical Modeling of *TAS* Delay

As illustrated in Figure 3, the operation cannot be executed when the target data frame arrives as the interfering frame is being transmitted. Once the transmission is finished, the interfering frame is inserted into the protection window, during which non-CDT data frames cannot be transmitted. Therefore, the time delay caused by the *TAS* mechanism is calculated as follows:

$$R_{TAS} = L_P = L_{CDT} + L_{GB} \tag{5}$$

where L_P is the frame length of the protection window, L_{CDT} is the frame length of the *CDT* time slot, and L_{GB} is the frame length of the protection band.

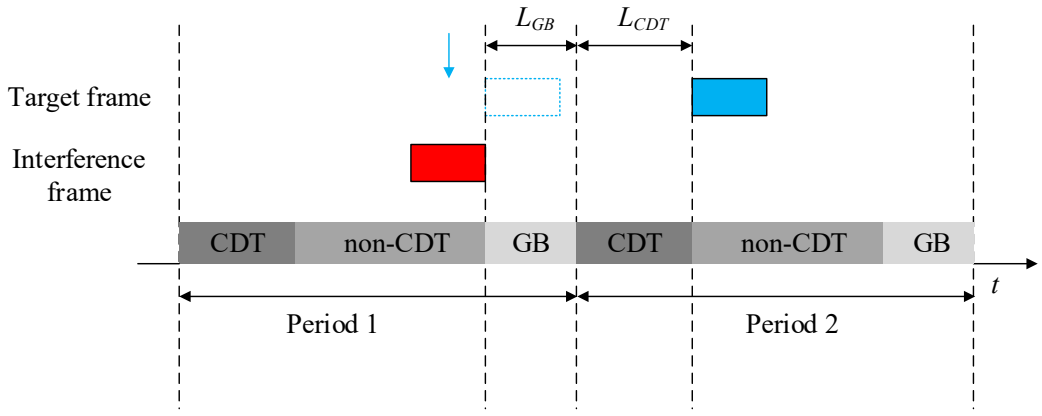


Figure 3. Process of forwarding data frames by *TAS* mechanism.

2.2. Delay Modeling

2.2.1. Class *A* Data Delay Modeling

Following the aforementioned analysis, the delay of Class *A* data comprises four types of time delays, namely *FIFO*, *CBS*, *TAS*, and forwarding delay [30], i.e.,

$$R_A = R_{FIFO_A} + R_{CBS_A} + R_{TAS_A} + R_{T_A} \tag{6}$$

In accordance with the *FIFO* algorithm, Class *A* data frames cannot be transmitted when they enter the queue due to the presence of interfering frames. These interfering frames are transmitted with unknown conditions and varying delays. As a result, the unfinished transmission portion can be regarded as an equivalent frame. The *FIFO* delay of a Class *A* data frame is represented by the following equation:

$$R_{FIFO_A} = \frac{M'_\tau}{C}\left(1 + \frac{sendSlope_A}{idleSlope_A}\right) \tag{7}$$

where: $sendSlope_A$ is the rate of credit decline for Class *A*, $idleSlope_A$ is the rate of credit growth for Class *A*, and M'_τ is the equivalent frame length.

Within the context of the *CBS* algorithm, time delays can be classified into two categories, namely low and high priority traffic blocking delays. As Class *A* data frames are assigned the highest priority, only low-priority delays may occur [31]. If the transmission of a low-priority data frame disrupts the transmission of a Class *A* data frame, then the completion of the latter is uncertain, and the length of the unsent segment can be regarded as the equivalent frame length. Therefore, the *CBS* delay for a Class *A* data frame is expressed as follows:

$$R_{CBS_A} = {t^A}_{LPB} = \frac{M'_L}{C}, M'_L = M'_B \ or \ M'_{BE} \tag{8}$$

where ${t^A}_{LPB}$ is the L-class blocking delay for Class *A* frames under the *CBS* algorithm, and M'_L is the equivalent frame length for *L*-class interference frame blocking under the *CBS* algorithm.

Within the *TAS* mechanism, the portion of the protection window that cannot be transmitted due to the arrival of Class *A* data can be considered as the equivalent length. Consequently, the *TAS* delay for a Class *A* data frame is expressed as follows:

$$R_{TAS_A} = L'_P \tag{9}$$

where L'_P is the equivalent length.

In summary, the Class *A* data delay is equal to the following:

$$\begin{aligned} R_A &= R_{FIFO_A} + R_{CBS_A} + R_{TAS_A} + R_{T_A} \\ &= \frac{M'_\tau}{C}\left(1 + \frac{sendSlope_A}{idleSlope_A}\right) + \frac{M'_L}{C} + L'_P + \frac{M_A}{C} \\ &= \frac{M_A + M'_L}{C} + \frac{M'_\tau}{C}\left(1 + \frac{sendSlope_A}{idleSlope_A}\right) + L'_P \end{aligned} \tag{10}$$

where M_A is the frame length of the Class *A* target data frame.

2.2.2. Class *B* Data Delay Modeling

Similarly, the queue forwarding delay expression for Class *B* data frames is

$$R_B = R_{FIFO_B} + R_{CBS_B} + R_{TAS_B} + R_{T_B} \tag{11}$$

The expression for the queue forwarding delay under the *FIFO* algorithm is not repeated as it follows the same principle as that of Class *A* data frames.

In the *CBS* algorithm, Class *B* data frames have a lower priority compared to Class *A* and a higher priority compared to Class *BE*. This leads to the coexistence of traffic blocking delays with smaller priorities and higher traffic blocking delays. The completion of transmission for Class *BE* interfering frames is uncertain, and the remaining portion can be designated as the equivalent length. Hence, the *CBS* delay for Class *B* interference frames can be expressed as follows:

$$t^B{}_{LPB} = \frac{M'_{BE}}{C} \tag{12}$$

where $t^B{}_{LPB}$ is the *L*-level blocking delay of Class *B* data frames under the *CBS* algorithm, and M'_{BE} is the equivalent frame length of the blocking of Class *BE* interference frames under the *CBS* algorithm.

Similarly, the high-priority traffic blocking for Class *B* data frames can be obtained as shown in the following equation:

$$t^B{}_{HPB} = R_A - t^A{}_{LPB} - R_{TAS_A} = R_{FIFO_A} + R_{T_A} \tag{13}$$

where $t^B{}_{HPB}$ is the *H*-level blocking delay for Class *B* data frames under the *CBS* algorithm.

So, the *CBS* delay for Class *B* data frames is

$$R_{CBS_B} = t^B{}_{LPB} + t^B{}_{HPB} = \frac{M'_{BE}}{C} + R_{FIFO_A} + R_{T_A} \tag{14}$$

The Class *B* delay under the *TAS* algorithm can be coterminous with Class *A*, and therefore will not be repeated, i.e.,

$$R_{TAS_B} = L'_P \tag{15}$$

In summary, the Class *B* data delay is equal to the following:

$$\begin{array}{c} R_B = R_{FIFO_B} + R_{CBS_B} + R_{TAS_B} + R_{T_B} \\ = \frac{M'_\tau}{C}\left(1 + \frac{sendSlope_B}{idleSlope_B}\right) + \frac{M'_{BE}}{C} + L'_P + \frac{M_B}{C} + R_{FIFO_A} + R_{T_A} \\ = \frac{M_B + M'_{BE}}{C} + \frac{M'_\tau}{C}\left(1 + \frac{sendSlope_B}{idleSlope_B}\right) + R_{FIFO_A} + R_{T_A} + L'_P \end{array} \tag{16}$$

where M_B is the frame length of the Class *B* target data frame.

2.2.3. Modeling of Latency for *BE* Class Data

Similarly for Class *A* and *B*, the queue forwarding delay for Class *BE* data frames can be expressed as

$$R_{BE} = R_{FIFO_{BE}} + R_{CBS_{BE}} + R_{TAS_{BE}} + R_{T_{BE}} \tag{17}$$

Under the *FIFO* algorithm, the Class *BE* is congruent to the Class *A* and *B*. Since there is no credit value for the Class *BE* data stream, the *FIFO* delay of the Class *BE* data frame is shown in the following equation:

$$R_{FIFO_{BE}} = \frac{M'_\tau}{C} \tag{18}$$

Under the *CBS* algorithm, only a high-priority traffic blocking delay exists for Class *BE* frames. If a Class *BE* data frame is blocked by a high-priority interfering frame, it is uncertain how much of the frame has been transmitted, and the remaining portion can be designated as the equivalent frame length. The *CBS* delay of a Class *BE* data frame can be expressed using the following equation:

$$R_{CBS_{BE}} = t^{BE}{}_{LPB} = \frac{M'_H}{C}, M'_H = M'_A \; or \; M'_B \tag{19}$$

where ${t^{BE}}_{LPB}$ is the H-level blocking delay of the BE class data frame under the CBS algorithm, and ${M'}_H$ is the blocking equivalent length of the class H interfering frame under the CBS algorithm.

Under the TAS algorithm, the treatment of Class BE frames is similar to that of Classes A and B. Thus, the TAS delay for Class BE data frames can be expressed using the following equation:

$$R_{TAS_{BE}} = {L'}_P \tag{20}$$

In summary, the queue forwarding delay of a BE class data frame can be expressed as

$$\begin{aligned} R_{BE} &= R_{FIFO_{BE}} + R_{CBS_{BE}} + R_{TAS_{BE}} + R_{T_{BE}} \\ &= \frac{{M'}_\tau}{C} + \frac{{M'}_H}{C} + {L'}_P + \frac{M_{BE}}{C} \\ &= \frac{M_{BE}+{M'}_H+{M'}_\tau}{C} + {L'}_P \end{aligned} \tag{21}$$

3. Forwarding Strategy Model Optimization

3.1. SARSA Reinforcement Learning Theory

SARSA reinforcement learning is a network synchronous learning method based on the temporal difference algorithm [32–35]. This approach updates state–action values using feedback information from the environment, without requiring knowledge of the state transfer, environmental model, or behavioral strategy. Consequently, the convergence speed of the algorithm can be significantly accelerated. The algorithmic procedure for SARSA reinforcement learning is as follows:

(1) Initialize the Q-value table.
(2) Repeat the following steps: In states, use the greedy algorithm to select action a based on the Q-value table. Obtain the reward r and transition to the next states. Update the Q-value table using the iterative equation.
(3) Output the optimal policy.

The selection of action a and the iterative formula for updating the Q-value table are crucial components of the entire process. The ε-greedy algorithm is the strategy employed by the SARSA algorithm for selecting actions. This strategy allows the intelligent agent to randomly choose an action with a probability of ε, and choose the current optimal action with a probability of $1-\varepsilon$, aiming to achieve the global optimum. The specific algorithm is as follows:

$$\pi^{\varepsilon}(s,a) = \begin{cases} \pi(s,a) & ,1-\varepsilon \\ random\ from\ A & ,\varepsilon \end{cases} \tag{22}$$

$$\pi(s,a) = argmaxQ(s,a) \tag{23}$$

As the SARSA algorithm updates the Q-value table based on the next actual action taken, the iterative equation for the Q-value table is as follows:

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a)] \tag{24}$$

where α is the learning rate and γ is the reward discount.

3.2. Q-Table Determination

The action space A of the model can be quantized as $A = [1,2,3,4,5,6]$, where 1–6 are idle, entering protection window, transmitting Class A frames, transmitting Class B frames, transmitting Class BE frames, and maintaining the current transmission action [36], respectively.

The states of each type of data flow are taken as state quantities in the state space S. The state space S is as follows:

$$S = \begin{bmatrix} Credit_P & Amount_P & Transmit_P \\ Credit_A & Amount_A & Transmit_A \\ Credit_B & Amount_B & Transmit_B \\ Credit_{BE} & Amount_{BE} & Transmit_{BE} \end{bmatrix} \tag{25}$$

The credit value of Class A and B data streams is an indefinite value, while Class BE data streams and protection windows do not have credit values. However, in order to maintain consistency, the credit values of Class BE data streams and protection windows are consistently set to 0, thus establishing the state space, i.e., $Credit_{BE} \equiv 0$, and $Credit_p \equiv 0$. The state space S is partitioned according to the number of states in the state space, and the results are shown in Table 1.

Table 1. Status classification table.

s	AP	AA	CA	AB	CB	ABE	a
1	$\neq 0$	random	random	random	random	random	2
2	=0	$\neq 0$	≥ 0	$\neq 0$	≥ 0	$\neq 0$	3
3	=0	$\neq 0$	≥ 0	random	<0	$\neq 0$	3
6	=0	$\neq 0$	≥ 0	$\neq 0$	≥ 0	=0	3
7	=0	$\neq 0$	≥ 0	=0	≥ 0	=0	3
8	=0	$\neq 0$	<0	$\neq 0$	≥ 0	$\neq 0$	4
25	=0	=0	<0	$\neq 0$	≥ 0	=0	4
26	=0	=0	<0	random	<0	=0	1
27	=0	=0	<0	=0	≥ 0	=0	1
	TP		**TA**	**TB**		**TBE**	
28	0		0	0		1	6
29	0		0	1		0	6
30	0		1	0		0	6
31	1		0	0		0	6

In order to facilitate the analysis, the state S is divided according to the relationship of each variable with 0, and then the similar results are combined, and the results obtained are shown in Table 1.

Each state is assigned a reward value when it performs an action, with the highest reward value given to the best action. The maximum reward is set to 10, while the remaining actions are assigned a reward value of 0.

After performing each action, the state of the environment is updated thus iterating the Q-table, thus allowing the intelligent body to keep performing actions to accomplish learning. In this paper, $\alpha = 0.1$ and $\gamma = 0.9$ are set to obtain the final Q-table.

3.3. Calculation of Time Delays

After initializing the Q-value table and states, the greedy algorithm filters and executes the action a, followed by updating the state space parameters and calculating the time delay value to update the states. This process is cyclically executed until $t < 10,000,000$. Afterward, the average time delay value is calculated for each type of data. The computational flowchart of SARSA reinforcement learning is illustrated in Figure 4.

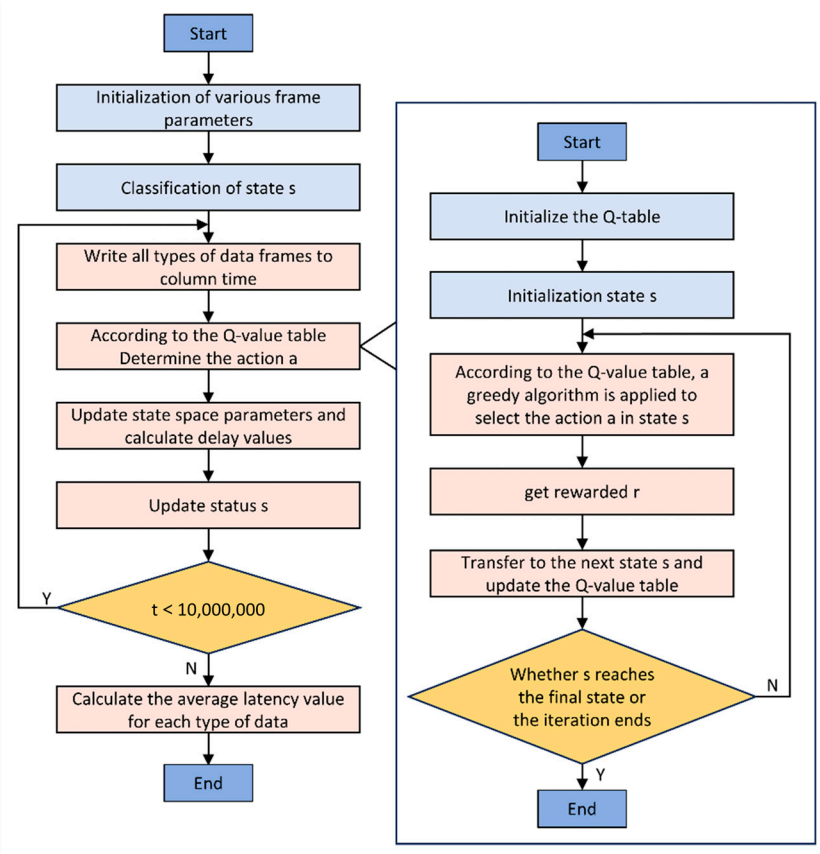


Figure 4. Flow of delay calculation based on SARSA reinforcement learning.

4. Simulation and Result Analysis

The data flow at the gateway of an in-vehicle TSN network usually belongs to the periodic traffic that helps to calculate the delay value. The delay values of different types of data frames are calculated so as to verify the reasonableness of the model proposed in this paper.

Referring to previous studies [37], the total link bandwidth is set to 100 Mbit/s, each gating period is set to 500 µs, and the non-CDT transmission bandwidth is set to 90 Mbit/s. where the CDT time slot length and the non-CDT time slot length are set to 150 µs and 350 µs, respectively. the lengths of the data frames of Class *A*, *B*, and *BE* are all set to 400 bytes. The Class *A* data frame period and Class *B* data frame period are set to 125 µs and 250 µs, respectively, and the reserved bandwidth of the data stream is set according to the actual situation. The reserved bandwidths for the Class *A* and Class *B* data streams are 400 Mbit/s and 20 Mbit/s, respectively.

The corresponding program is run in MATLAB-R2020b and the results are shown in Figures 5–7. Firstly, we compare and analyze the mean values of the delay of the data frame group of the SR class data frames under AVB and TSN networks, respectively, so that the parameters of the SR class data frames remain unchanged. Figure 5 shows the delay averages of Class *A* data frames in both TSN and AVB network environments when the BE load is 5 Mbit/s. It can be seen that although the protection window in TSN guarantees the transmission delay of CDT data frames, it has a great impact on the queue forwarding of SR class data frames, which is in line with the theoretical situation.

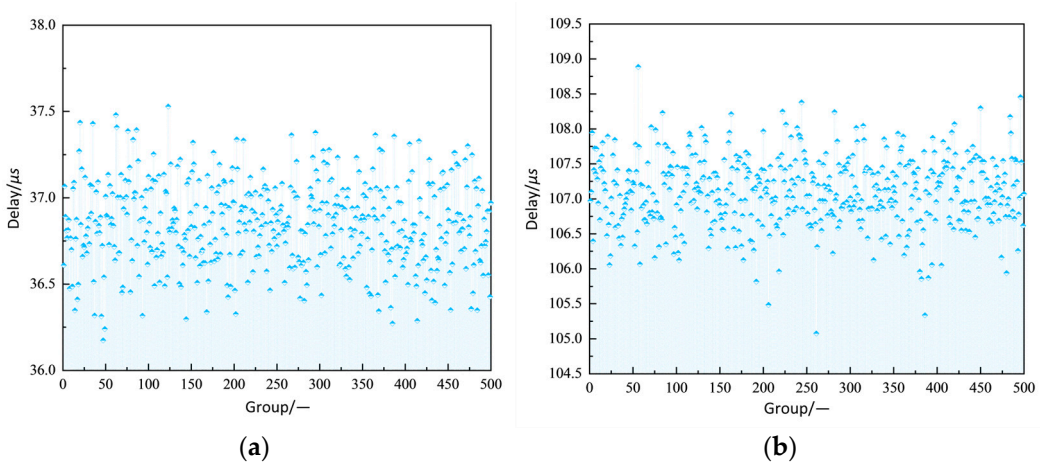


Figure 5. Comparison of AVB and TSN network delay averages. (**a**) Delay averages of Class *A* data frames in AVB network. (**b**) Delay averages of Class *A* data frames in TSN network.

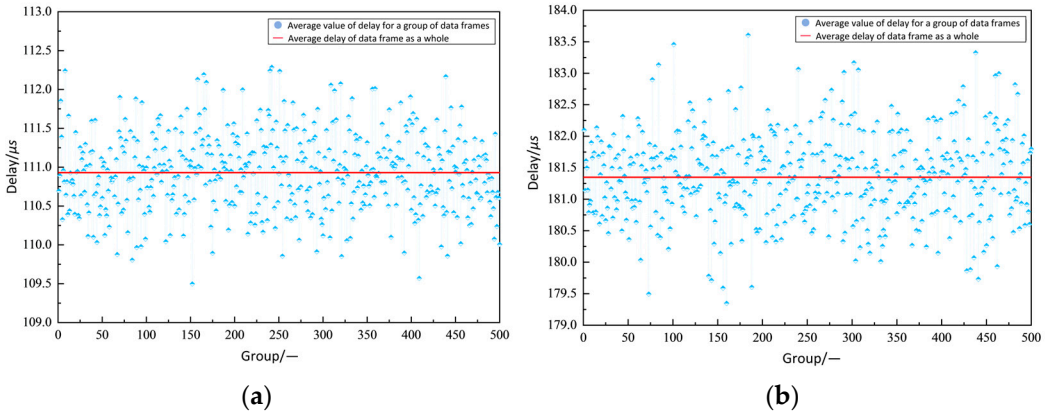


Figure 6. Graph of average delay statistics for traffic flow. (**a**) Class *A* data latency. (**b**) Class *B* data latency.

Secondly, the data delay of the two SR class data frames is compared and analyzed in the TSN network. Figure 6 shows the comparison of the average delay of traffic for Class *A* data frames and Class *B* data frames when the BE class load is 10 Mbit/s. The results are 110.93 μs and 181.35 μs, respectively, and also the discrete situation of Class *A* data frames and Class *B* data frames can be seen from the figure, i.e., the Class *A* mean square deviation is smaller than that of Class *B*. The results show that the transmission real time of Class *A* data frames is prioritized higher in TSN networks, i.e., Class *A* traffic is prioritized higher than Class *B*, which is in line with the previous data type analysis.

Next, the BE data load is analyzed to see if there is any significant effect on the delay of SR class data. As shown in Figure 7, Class *A* data frames are taken as the object of study, and the delay values of Class *A* data frames under different BE data frame loads are simulated and analyzed. It can be seen that the transmission delay of Class *A* data frames increases significantly with increasing load.

In summary, the TSN data forwarding delay analysis model developed in this paper matches the theoretical situation, and the fitting performance for the above scenarios is also better, thus verifying the reasonableness of the model.

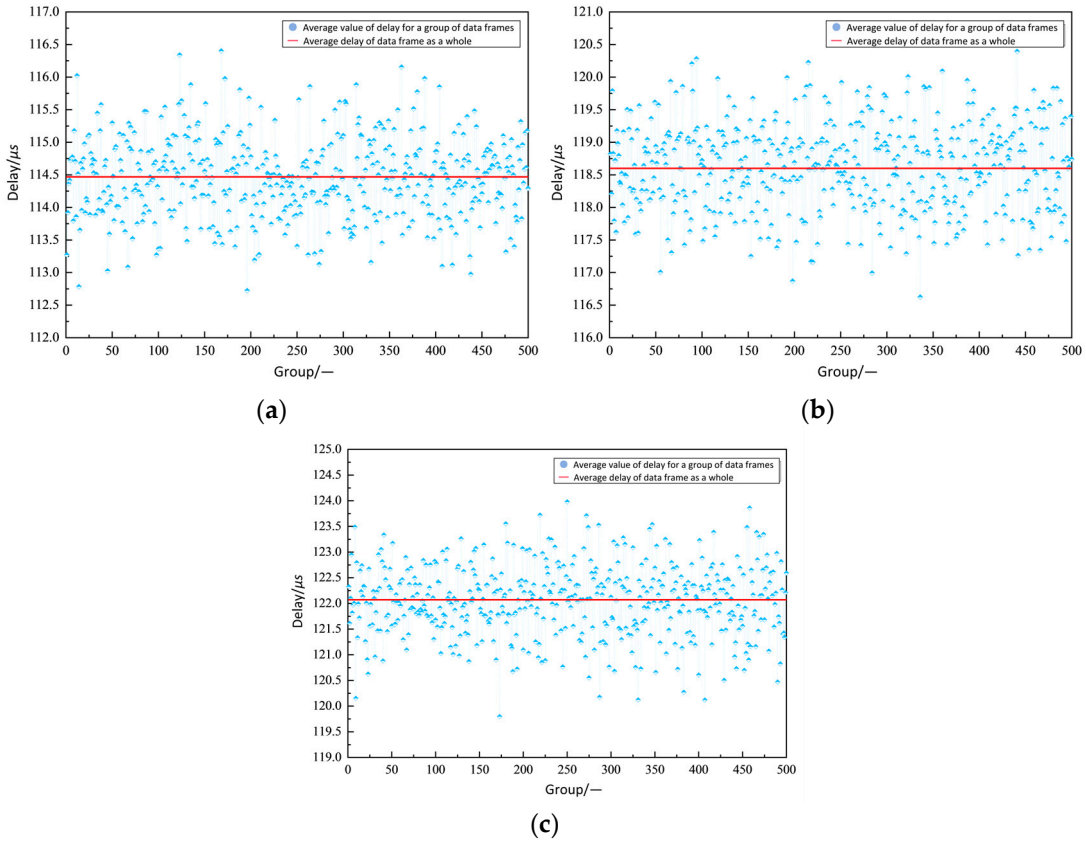


Figure 7. Delay statistics for Class *A* frames with different BE loads. (**a**) BE is loaded with 15 Mbit/s. (**b**) BE is loaded with 20 Mbit/s. (**c**) BE is loaded with 25 Mbit/s.

5. Test Validation

The simulation testbed consists of RAD_Galaxy, gateway, surround view system and PC as shown in Figure 8. The video data are captured by the four cameras of the surround view system, and the data are transmitted to the gateway at the same time. RAD_Galaxy acts as an intermediary device connecting the gateway to the PC and the monitor, and also acts as a sending node for the audio data and control data. This allows the data messages at the gateway to be monitored at the PC, thus realizing a semi-physical simulation; the test bench is shown in Figure 9.

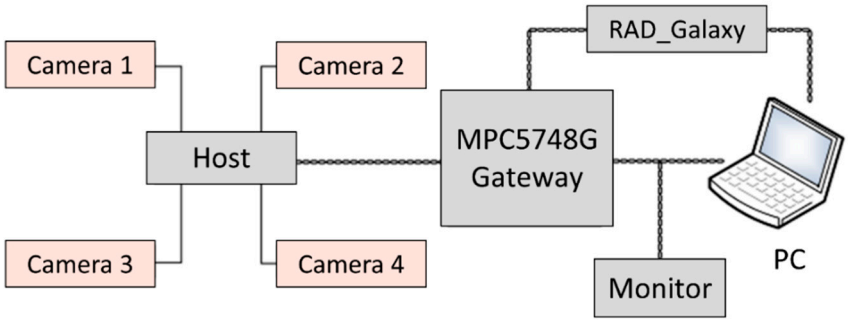


Figure 8. Schematic diagram of the test bench.

The experiment concludes by comparing the theoretical delay, obtained through simulation in MATLAB, with the actual measured delay values on the experimental bench. This comparison aims to verify the reasonableness of the TSN delay analysis model developed in this paper.

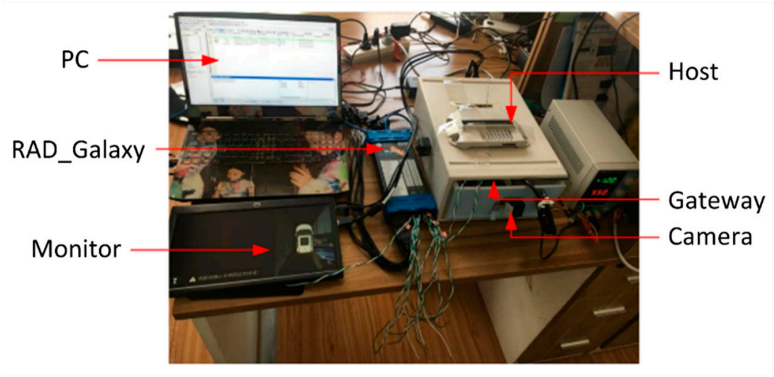


Figure 9. Physical drawing of the lab bench.

Due to the large variance of the individual time delay data of the measured samples, the average of the time delays was used as a comparison reference, with 40 data frames per subgroup. The results of the time delay comparison are shown in Figures 10 and 11. The results show that the delay values obtained from both model predictions and experimental measurements increase with the increase in Class *BE* data load, and the deviation values of both have a small increase. The theoretical and actual measured delay values for Class *A* and Class *B* data frames are shown in Tables 2 and 3. The deviation of both the theoretical and measured values of the reserved class data transmission delay is less than 5% for different BE data loads, with a maximum deviation of 4.64%.

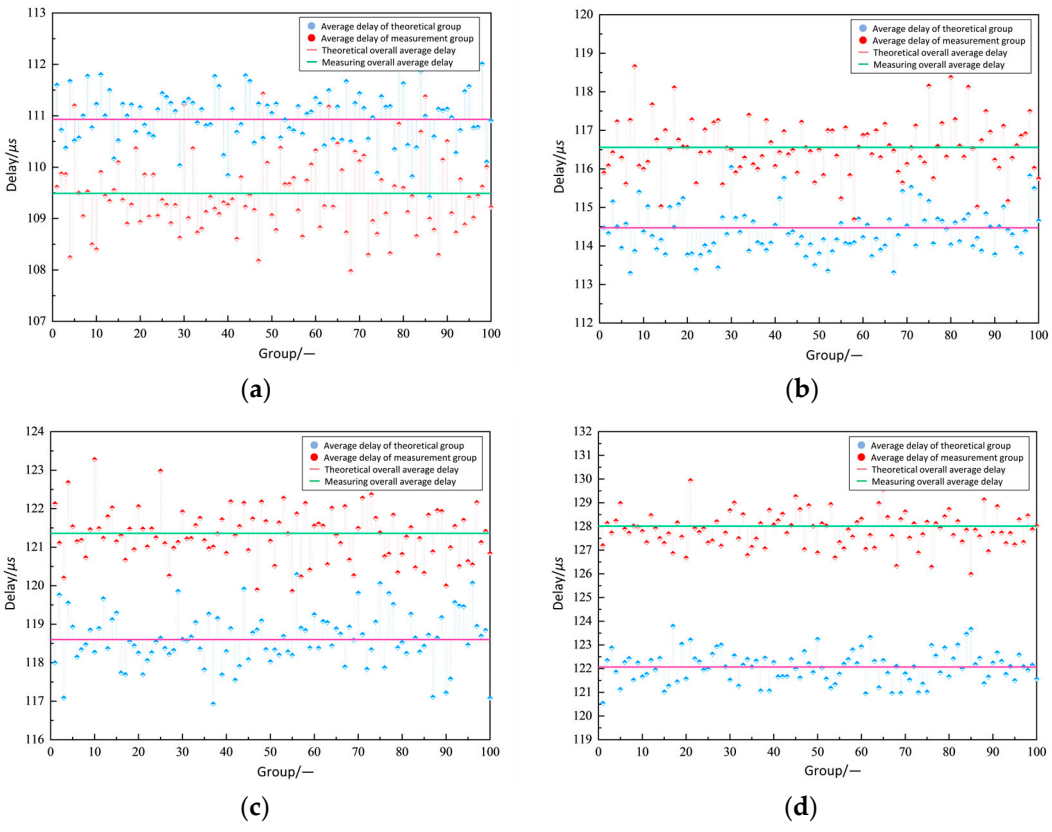


Figure 10. Comparison of theoretical and measured delay for Class *A* data frames. (**a**) BE is loaded with 10 Mbit/s. (**b**) BE is loaded with 15 Mbit/s. (**c**) BE is loaded with 20 Mbit/s. (**d**) BE is loaded with 25 Mbit/s.

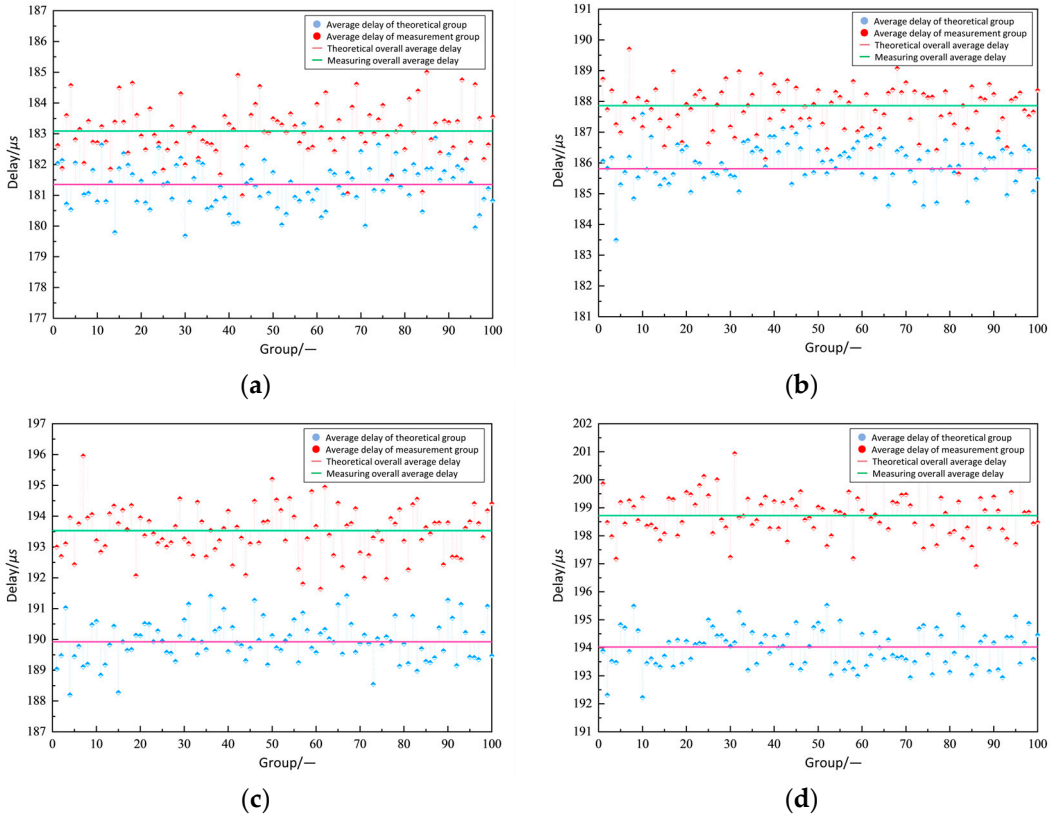


Figure 11. Comparison of theoretical and measured latency for Class *B* data. (**a**) BE is loaded with 10 Mbit/s. (**b**) BE is loaded with 15 Mbit/s. (**c**) BE is loaded with 20 Mbit/s. (**d**) BE is loaded with 25 Mbit/s.

Table 2. Theoretical and measured delay results for Class *A* data.

	Class *A* (Mbps)	Class *B* (Mbps)	Class *BE* (Mbps)	Theoretical Delay (μs)	Measured Delay (μs)	Deviation (%)
1	50	25	10	110.93	109.35	−1.44
2	50	25	15	114.47	115.91	1.24
3	50	25	20	118.60	122.76	3.39
4	50	25	25	122.07	128.43	4.95

Table 3. Theoretical and measured delay results for Class *B* data.

	Class *A* (Mbps)	Class *B* (Mbps)	Class *BE* (Mbps)	Theoretical Delay (μs)	Measured Delay (μs)	Deviation (%)
1	50	25	10	181.35	183.15	0.98
2	50	25	15	185.81	187.97	1.15
3	50	25	20	189.94	193.61	1.90
4	50	25	25	194.03	198.38	2.19

Furthermore, in order to validate the feasibility of the method, Table 4 is included, which presents a comparison of delay results using the Q-reinforcement learning approach. Through this comparative test, it becomes apparent that the delay analysis model put forth in this paper successfully captures the delay characteristics of reservation classes in TSN networks.

Table 4. Delay results based on Q reinforcement learning.

	Class *A* (Mbps)	Class *B* (Mbps)	Class *BE* (Mbps)	Theoretical Delay (μs)	Measured Delay (μs)	Deviation (%)
1	50	25	10	177.84	181.36	1.94
2	50	25	15	179.48	183.95	2.43
3	50	25	20	183.40	189.34	3.14
4	50	25	25	187.64	196.26	4.39

6. Conclusions

In this paper, we propose a TSN data forwarding delay model using the SARSA reinforcement learning algorithm for simulation studies. Building upon the *FIFO*, *CBS*, and *TAS* delay models, we establish delay models for both Class *A* and Class *B* data. To optimize the queue forwarding delay model, we utilize the SARSA reinforcement learning algorithm. We then verify the validity of our simulation model through simulations and physical object experiments, ensuring its accuracy and reliability.

Author Contributions: Conceptualization, C.H. and Y.W.; methodology, C.H.; software, Y.W.; validation, C.H. and Y.W.; formal analysis, C.H., Y.W. and Y.Z.; investigation, Y.W.; resources, Y.Z.; data curation, Y.W.; writing—original draft preparation, Y.W.; writing—review and editing, C.H.; visualization, Y.Z.; supervision, Y.Z.; project administration, C.H.; funding acquisition, Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the following grants: Open Fund Project of State Key Laboratory of Automotive Simulation and Control [20201206], China Postdoctoral Science Foundation Project [2021M691847], and Open Fund of State Key Laboratory of Automotive Safety and Energy Conservation [KFY2207].

Data Availability Statement: Data are contained within the article.

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

References

1. Hu, B.; Qin, G.; Liu, Y.; He, Y.; Wu, X. Next-generation automotive networks: Status and development of in-vehicle Ethernet technology. *Comput. Eng. Appl.* **2016**, *52*, 29–36.
2. Li, W.; Zhang, L.; Wang, Y. In-vehicle Ethernet technology and standardization. *Telecommun. Netw. Technol.* **2016**, *6*, 1–5.
3. Farkas, J.; Bello, L.L.; Gunther, C. Time-Sensitive Networking Standards. *IEEE Commun. Stand. Mag.* **2018**, *2*, 20–21. [CrossRef]
4. *IEEE Std 802. 1QatTM-2010*; IEEE Standard for Local and Metropolitan Area Networks Virtual Bridged Local Area Networks Amendment 14: Stream Reservation Proticol (SRP). IEEE: Piscataway, NJ, USA, 2010.
5. *IEEE Std 802. 1ASTM-2011*; IEEE Standard for Local and Metropolitan Area Networks—Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks. IEEE: Piscataway, NJ, USA, 2011.
6. *IEEE 802. 1Qav-2009*; Standard for Local and Metropolitan Area Networks-Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams. IEEE: Piscataway, NJ, USA, 2009.
7. *IEEE 802. 1Qbv-2015*; Standard for Local and Metropolitan Area Networks-bridges and Bridged Networks-Amendment 25: Enhancements for Scheduled Traffic. IEEE: Piscataway, NJ, USA, 2016.
8. *IEEE 802. 1Qbu-2016*; Standard for Local and Metropolitan Area Networks-Bridges and Bridged Networks-Amendment 26: Frame Preemption. IEEE: Piscataway, NJ, USA, 2016.
9. *IEEE P802.1Qch/D2.0.*; Draft Standard for Local and Metropolitan Area Networks-Bridges and Bridged Networks-Amendment: Cyclic Queuing and Forwarding. IEEE: Piscataway, NJ, USA, 2016.
10. *IEEE 1722-2016*; IEEE Computer Society: Transport Protocol for Time-Sensitive Applications in Bridged Local Area Networks. IEEE: New York, NY, USA, 2016.
11. *IEEE 802. 3br-2016*; Standard for Ethernet Amendment 5: Specification and Management Parameters for Interspersing Express Traffic. IEEE: Piscataway, NJ, USA, 2016.
12. Kirsten, M. *Automotive Ethernet*; Mechanical Industry Press: Beijing, China, 2019.
13. Hank, P.; Mü, S. Automotive Ethernet: In-vehicle Networking and Smart Mobility. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition*; IEEE: Manhattan, NY, USA, 2013.
14. Fu, S.; Zhang, H.; Chen, J. Time Sensitive Networking Technology Overview and Performance Analysis. *ZTE Commun.* **2018**, *16*, 57–64.

15. Bruckner, D.; Stănică, M.P.; Blair, R.; Schriegel, S.; Kehrer, S.; Seewald, M.; Sauter, T. An Introduction to OPC UA TSN for Industrial Communication Systems. *Proc. IEEE* **2019**, *99*, 1121–1131. [CrossRef]
16. Bello, L.L.; Steiner, W. A Perspective on IEEE Time-sensitive Networking for Industrial Communication and Automation Systems. *Proc. IEEE* **2019**, *107*, 1094–1120. [CrossRef]
17. Cao, Z. *Research on Data Delay Modeling for In-Vehicle TSN Reservation Class Data*; Jiangsu University: Zhenjiang, China, 2022.
18. Imtiaz, J.; Jasperneite, J.; Han, L. A Performance Study of Ethernet Audio Video Bridging (AVB) for Industrial Real-time Communication. In *Proceedings of the IEEE Conference on Emerging Technologies & Factory Automation*; IEEE: Manhattan, NY, USA, 1946.
19. Maxim, D.; Song, Y.Q. Delay Analysis of AVB traffic in Time-sensitive networks (TSN). In Proceedings of the RTNS 2017—International Conference on Real-Time Networks and Systems, Grenoble, France, 4–6 October 2017.
20. Thangamuthu, S.; Concer, N.; Cuijpers, P.J.; Lukkien, J.J. Analysis of Ethernet-Switch Traffic Shapers for In-Vehicle Networking Applications. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition*; IEEE: Manhattan, NY, USA, 2015.
21. Thiele, D.; Ernst, R.; Diemer, J. Formal Worst-case Timing Analysis of Ethernet TSN's Time-aware and Peristaltic shapers. In Proceedings of the 2015 IEEE Vehicular Networking Conference (VNC), Kyoto, Japan, 16–18 December 2016; IEEE: Manhattan, NY, USA, 2016.
22. Zhao, L.; Pop, P.; Zheng, Z.; Li, Q. Timing Analysis of AVB Traffic in TSN Networks Using Network Calculus. In Proceedings of the 2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), Porto, Portugal, 11–13 April 2018.
23. Mohammadpour, E.; Stai, E.; Mohiuddin, M.; Le Boudec, J.Y. End-to-End Latency and Backlog Bounds in Time-sensitive Networking with Credit Based Shapers and Asynchronous Traffic Shaping. In Proceedings of the 2018 30th International Teletraffic Congress (ITC 30), Vienna, Austria, 3–7 September 2018; IEEE: Manhattan, NY, USA, 2018.
24. Wang, Y.; Huang, F.; Li, Y.; Pan, B.; Wu, Y. Hierarchical Scheduling and Real-time Analysis for vehicular Time-sensitive Network. In Proceedings of the 2019 12th International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, China, 14–15 December 2019; IEEE: Manhattan, NY, USA, 2019.
25. Xu, H.-L.; Gao, H.; Wang, P.; Zhang, H.; Cao, W. A hybrid probabilistic model for frame preemption delay in time-sensitive networks. *J. North China Electr. Power Univ. (Nat. Sci. Ed.)* **2023**, *50*, 100–108.
26. Hellmanns, D.; Glavackij, A.; Falk, J.; Hummen, R.; Kehrer, S.; Dürr, F. Scaling TSN Scheduling for Factory Automation Networks. In Proceedings of the 16th IEEE International Conference on Factory Communication Systems, Porto, Portugal, 27–29 April 2020; IEEE: Manhattan, NY, USA, 2020.
27. Craciunas, S.S.; Oliver, R.S.; Chmelík, M.; Steiner, W. Scheduling Real-Time Communication in IEEE 802.1Qbv Time Sensitive Networks. In *Proceedings of the International Conference on Real-Time Networks & Systems*; ACM: New York, NY, USA, 2016.
28. Ciarletta, L.; Fejoz, L.; Guenard, A.; Navet, N. Development of a safe CPS component: The hybrid parachute, a remote termination add-on improving safety of UAS. In Proceedings of the Embedded Real-Time Software and Systems (ERTS 2016), Toulouse, France, 27–29 January 2016.
29. Navet, N.; Fejoz, L.; Havet, L.; Sebastian, A. Lean Model-Driven Development through Model-Interpretation: The CPAL design flow. In Proceedings of the Embedded Real-Time Software and Systems (ERTS 2016), Toulouse, France, 27–29 January 2016.
30. Altmeyer, S.; Navet, N.; Fejoz, L. Using CPAL to model and validate the timing behaviour of embedded systems. In Proceedings of the 6th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS), Lund, Sweden, 7 July 2015.
31. Seyler, J.; Navet, N.; Fejoz, L. Insights on the Configuration and Performances of SOME/IP Service Discovery. In Proceedings of the SAE World Congress, Detroit, MI, USA, 21–23 April 2015.
32. Peng, Q. *Research on TSN Scheduling Algorithm for Industrial Internet of Things*; Zhejiang University of Technology: Hangzhou, China, 2019.
33. Wang, Y. *Research on Online Scheduling Algorithm for Real-Time Streams in Time-Sensitive Networks*; Dalian University of Technology: Dalian, China, 2021.
34. Zhou, X.; Qiu, Z.; Zhu, J.; Zhang, H.; Zhao, C. A GCL adaptive adjustment algorithm for time-sensitive networks. *J. Beijing Univ. Posts Telecommun.* **2022**, *45*, 26–32. [CrossRef]
35. Qin, Z.; Li, N.; Liu, X.; Liu, X.; Tong, Q. A review of model-free reinforcement learning. *Computer Science* **2021**, *48*, 180–187.
36. Tesauro, G. Temporal Difference Learning and TD-Gammon. *Commun. ACM* **1995**, *38*, 58–68. [CrossRef]
37. Hong, J. *Mathematical Modeling of Delay Characteristics of AVB Data Transmission over Vehicular Ethernet*; Jiangsu University: Zhenjiang, China, 2020.

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.