electronics

MDPI

Article

Optimizing Traffic Scheduling in Autonomous Vehicle Networks Using Machine Learning Techniques and Time-Sensitive Networking

Ji-Hoon Kwon [1], Hyeong-Jun Kim [2,*] and Suk Lee [1,*]

[1] School of Mechanical Engineering, Pusan National University, Busan 46241, Republic of Korea; jhkwon09@pusan.ac.kr
[2] Department of Future Automotive Engineering, Gyeongsang National University, Jinju 52725, Republic of Korea
* Correspondence: hj.kim@gnu.ac.kr (H.-J.K.); slee@pnu.edu (S.L.)

Abstract: This study investigates the optimization of traffic scheduling in autonomous vehicle networks using time-sensitive networking (TSN), a type of deterministic Ethernet. Ethernet has high bandwidth and compatibility to support various protocols, and its application range is expanding from office environments to smart factories, aerospace, and automobiles. TSN is a representative technology of deterministic Ethernet and is composed of various standards such as time synchronization, stream reservation, seamless redundancy, frame preemption, and scheduled traffic, which are sub-standards of IEEE 802.1 Ethernet established by the IEEE TSN task group. In order to ensure real-time transmission by minimizing end-to-end delay in a TSN network environment, it is necessary to schedule transmission timing in all links transmitting ST (Scheduled Traffic). This paper proposes network performance metrics and methods for applying machine learning (ML) techniques to optimize traffic scheduling. This study demonstrates that the traffic scheduling problem, which has NP-hard complexity, can be optimized using ML algorithms. The performance of each algorithm is compared and analyzed to identify the scheduling algorithm that best meets the network requirements. Reinforcement learning algorithms, specifically DQN (Deep Q Network) and A2C (Advantage Actor-Critic) were used, and normalized performance metrics (E2E delay, jitter, and guard band bandwidth usage) along with an evaluation function based on their weighted sum were proposed. The performance of each algorithm was evaluated using the topology of a real autonomous vehicle network, and their strengths and weaknesses were compared. The results confirm that artificial intelligence-based algorithms are effective for optimizing TSN traffic scheduling. This study suggests that further theoretical and practical research is needed to enhance the feasibility of applying deterministic Ethernet to autonomous vehicle networks, focusing on time synchronization and schedule optimization.

Keywords: reinforcement learning; advantage actor-critic (A2C); deep Q network (DQN); time sensitive networking (TSN); scheduling

check for updates

Citation: 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. https://doi.org/10.3390/electronics13142837

Academic Editor: Fabio Corti

Received: 10 June 2024
Revised: 8 July 2024
Accepted: 16 July 2024
Published: 18 July 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

Recent advancements in autonomous driving technology, satisfying level 3 and above as specified by the Society of Automotive Engineers (SAE), necessitate a higher level of safety and reliability. Level 3 autonomous vehicles operate autonomously using numerous advanced driver assistance system sensors and external infrastructure information [1]. However, the vast amount of data required for autonomous driving necessitates a higher network bandwidth than that provided by traditional vehicle network technologies such as the control area network and FlexRay [2,3]. In addition, the loss or latency of time-critical data in the network poses severe challenges for autonomous vehicle control, thereby

necessitating a vehicle network capable of transmitting such data reliably and in real-time [4–8]. Ethernet has emerged as a potential network technology capable of meeting these high demands.

Ethernet, a representative network technology, offers high bandwidth and compatibility with various protocols, extending its application from office environments to smart factories, aerospace, and the automotive industry [9]. Generally, Ethernet employs best-effort (BE) delivery, transmitting messages within a given bandwidth, regardless of frame importance or loss. However, the BE can result in frame loss or high latency under increased network traffic, which critically affects the real-time and reliability requirements of autonomous driving. This indicates the need for enhanced Ethernet technology to ensure stability and real-time delivery of time-critical traffic.

Deterministic Ethernet, which is designed to overcome the shortcomings of traditional Ethernet, ensures precise time synchronization among network devices, extremely low frame loss, and limited latency [10]. Time-sensitive networking (TSN) is a representative deterministic Ethernet technology comprising various IEEE 802.1 Ethernet sub-standards developed by the IEEE TSN task group, including time synchronization, stream reservation, seamless redundancy, frame preemption, and scheduled traffic (ST). ST based on time synchronization minimizes queue delays and guarantees bounded latency and real-time performance by transmitting frames at prescheduled times, even in congested networks.

IEEE 802.1 Qbv, the standard for ST, specifies the transmission mechanism for ST operating at egress ports [11]. According to IEEE 802.1 Qbv, network traffic resides in one of the eight egress port queues based on priority, with transmission determined by the queue state controlled by the gate control list (GCL) time scheduler. Another relevant standard, IEEE 802.1 Qbu, allows ST to be prioritized using frame preemption, even when other traffic occupies the bandwidth [12]. These standards enable autonomous driving networks to transmit high-priority control and sensor messages according to a predefined scheduler, thereby minimizing potential delays [13].

In a typical network environment, an ST traverses multiple links to reach its destination. Therefore, to minimize the end-to-end (E2E) delay of ST messages and guarantee real-time transmission, it is necessary to schedule the transmission timing on all links involved [14]. This task resembles the NP-hard complexity of job or flow scheduling problems, prompting various scheduling approaches to optimize deterministic Ethernet schedules [15].

Steiner proposed a scheduling technique using a satisfiability modulo theories (SMT) solver for time-triggered Ethernet (TT-Ethernet) to ensure ST traffic transmission [16]. Subsequent studies introduced slots between schedules to reduce delays in rate-constrained traffic [17]. Craciunas formulated a scheduling problem using logical constraints and proposed optimization methods based on SMT and maximum intensity projection [18,19]. Selicean suggested a tabu search-based metaheuristic to minimize the worst-case delay in TT traffic scheduling [20]. Gavriluţ applied a greedy randomized adaptive search procedure to schedule both TSN and AVB traffic for industrial applications [21]. Durr defined the scheduling of critical data as a no-wait job shop scheduling problem and proposed a schedule compression algorithm to reduce bandwidth waste caused by guard bands [22]. Chen focuses on the delay problem existing in the real-time data transmission process of in-vehicle Ethernet and innovatively proposes a fixed point message scheduling algorithm (FPMS) based on time-sensitive network (TSN) technology [23]. Zheng proposes the CSDN-TCS (Clock Synchronization Deterministic Network Traffic Classification Scheduling) algorithm, which aims to address how to schedule as many packets as possible under deadline constraints [24]. Huang proposes an intelligent SARSA (State–Action–Reward–State–Action) reinforcement learning algorithm for delay analysis of reservation class data in vehicular TSN networks [25]. While previous studies have focused on improving the transmission delay effect by using heuristic scheduling methods, this paper contributes by suggesting the possibility of methods that can be applied to machine learning techniques. In particular, while Huang's paper analyzed transmission delay using SARSA among

machine learning methods, this paper suggests the possibility of applying it to scheduling optimization by using DQN (deep Q network) and A2C (Advantage Actor-Critic).

This study proposes an algorithm to optimize the traffic schedule problem in a simulation environment by modeling a time-synchronized autonomous driving network using reinforcement learning. We define the traffic schedule problem as a sequential decision-making problem and design a reward function that maximizes the cumulative reward to improve the network performance. The schedule performance is evaluated using a cost function parameterized by three performance indicators: E2E delay, jitter, and bandwidth utilization for guard bands (BUGB). Each parameter is calculated as the weighted sum of the standardized performance metrics. Finally, we simulate an autonomous driving network to evaluate the performance of the proposed scheduling optimization algorithm and verify its applicability.

The remainder of this paper is organized as follows: Section 2 explains the network model for scheduling; Section 3 describes the MDP (Markov Decision Process) model for applying reinforcement learning; Section 4 discusses the reinforcement learning-based schedule performance evaluation environment and results; and Section 5 concludes the paper.

2. Network Modeling for Traffic Scheduling

2.1. Architecture Modeling

The architecture model, which abstractly represents the physical connections of the network, can be modeled as a directed graph $\mathcal{G}(E, V)$ and includes end systems (ES), switches (SW), and links. Here, V (vertices) refers to the devices constituting the network, which are ES and SW, and is defined as $V = ES \cup SW$. E (edges) represents the physical data links, where a link connecting v_a to v_b for $v_a, v_b \in V$ can be represented as $[v_a, v_b] \in E$. Each edge has three attributes: transmission rate, propagation delay, and queue number. The transmission rate, denoted by $[v_a, v_b].s$, represents the number of bits transmitted per second over the link. The propagation delay, denoted as $[v_a, v_b].d$, refers to the time required for an electrical signal to travel from the source to the destination through the physical medium. Finally, the queue number represents the number of queues present at the egress port, with the TSN standard allowing up to eight queues per egress port. The n-th queue of the egress port is represented by $[v_a, v_b].n$. Figure 1 illustrates a network topology comprising three ES and two SW. The transmission path (route) of the traffic is denoted by r_k, which is a list of edges traversed by the traffic from the source to the destination in order. In Figure 1, for r_1, the source is ES1, and the destination is ES3, and it can be represented as a sequence of edges: $r_1 = ([ES_1, SW_1], [SW_1, SW_2], [SW_1, ES_3], [SW_2, ES_3])$.

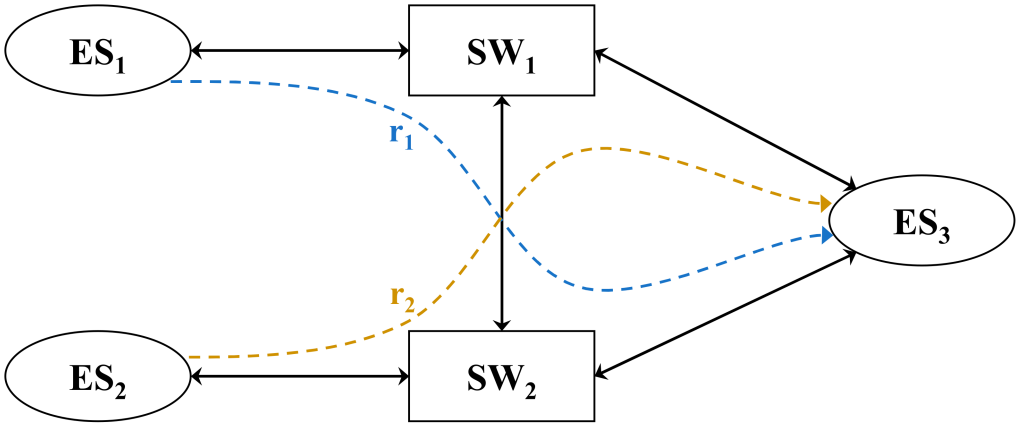


Figure 1. Simple TSN network topology for illustration.

2.2. Application Modeling

The application model represents the information regarding the messages required for TSN scheduling and the timing information necessary for the operation of the GCL,

denoted as $\mathcal{T}(S, C)$. The parameter S of the application model $\mathcal{T}$ represents the set of flows s_i, expressed as $s_i \in S$. Here, the flow s_i is an abstract representation of the messages periodically transmitted over a specific path, and in this paper, s_i is defined as the i-th ST. The transmission period of a flow is denoted as $s_i.T$, and the transmission path of the flow is expressed as $s_i.r_k$. Parameter C represents the factors affecting the open and closed states of the GCL, where the state of the GCL is determined by the delays occurring during frame transmission and the start of frame transmission.

A frame f is a unit of data exchanged in the Ethernet network, represented as $f \in F$, where F denotes the set of frames. The m-th frame of flow s_i transmitted through link $[v_a, v_b]$ is denoted as $f_{i,m}^{[v_a,v_b]}$, with the frame size expressed as $f_{i,m}^{[v_a,v_b]}.B$. Equation (1) represents the network delay $f_{i,m}^{[v_a,v_b]}.D$, which occurs when a frame is transmitted over link $[v_a, v_b]$, and is defined as the sum of the processing, queuing, transmission, and propagation delays. The processing delay, denoted as $f_{i,m}^{[v_a,v_b]}.p$, refers to the time required to check the header, determine the destination, and perform error checking when the frame is received at v_a. The queuing delay, denoted as $f_{i,m}^{[v_a,v_b]}.q$, refers to the temporary waiting time until the previously received frames are processed at the egress port of v_a. Transmission delay, denoted as $f_{i,m}^{[v_a,v_b]}.t$, is the frame size $f_{i,m}^{[v_a,v_b]}.B$ divided by the transmission rate $[v_a, v_b].s$.

$$f_{i,\ m}^{[v_a,\ v_b]}.D = f_{i,\ m}^{[v_a,\ v_b]}.p + f_{i,\ m}^{[v_a,\ v_b]}.q + f_{i,\ m}^{[v_a,\ v_b]}.t + [v_a,\ v_b].d \tag{1}$$

The E2E delay refers to the sum of all delays in the transfer of a frame from the source to the destination node. Since the E2E delay of each frame varies with the number of hops in the link through which the frame passes, an additional calculation process is required.

Equation (2) presents the E2E delay performance indicator. The term $f_{i,\ m}^{S_i.r}.D_{real}$ is the measured E2E delay when a frame is transmitted according to the schedule, and $f_{i,\ m}^{S_i.r}.D_{ideal}$ is defined as the E2E delay under the assumption that there is no queuing delay, i.e., there are no other frames when the frame is transmitted. Here, the E2E delay performance indicator was calculated as the sum of the ratios of the difference between the real $f_{i,\ m}^{S_i.r}.D_{real}$ and the ideal $f_{i,\ m}^{S_i.r}.D_{ideal}$ to $f_{i,\ m}^{S_i.r}.D_{ideal}$. The variable n refers to the number of flows, and k_i refers to the number of frames transmitted during the hyperperiod in the i-th flow.

$$E2E\ delay = \sum_{i=1}^{n}\{\sum_{m=1}^{k_i}(\frac{f_{i,m}^{s_i.r}.D_{real} - f_{i,m}^{s_i.r}.D_{ideal}}{f_{i,m}^{s_i.r}.D_{ideal}})\} \tag{2}$$

In a network environment, jitter refers to a phenomenon in which frame delays are inconsistent, also known as latency variation. In this study, the jitter is defined as the standard deviation of the time intervals at which frames are received at the destination during a hyperperiod. For frames transmitted only once during the hyperperiod, the jitter is defined as 0. Equation (3) shows the performance metric for the jitter, $Fitness_J$, which is calculated as the average jitter for all frames. Here, $\tau_{i,m}$ represents the difference in transmission completion times between the m-th and (m + 1)-th frames of flow s_i during the hyper period, while $\overline{\tau}$ denotes the average of $\tau_{i,m}$ measured over the hyper period. k_i represents the number of frames of flow s_i transmitted during the hyperperiod, and n denotes the number of flows s_i.

$$Jitter = \frac{1}{n}\sum_{i=1}^{n}\sqrt{\sum_{m=1}^{k_i-1}\left(\frac{(\tau_{i,m} - \overline{\tau})^2}{k_i}\right)} \tag{3}$$

IEEE 802.1 Qbv uses a guard band of maximum frame size (MFS, 1542 bytes) before transmitting an ST at a scheduled time to prevent non-ST frames from interfering with the transmission of scheduled ST frames [9]. This ensures that the critical frames have a bounded latency. The MFS can be reduced to 124 bytes through frame preemption, as

defined by IEEE 802.1 Qbu [12]. However, an increase in the number of guard bands results in network bandwidth loss, which necessitates a schedule that minimizes the number of guard bands. Equation (4) presents the performance metric for BUGB, denoted as $Fitness_{GB}$, which is the ratio of the total transmission time of all guard bands on a link to the hyper period. In Equation (4), $l \in E$ represents the links constituting the network, $GB.cnt_l$ is the number of guard bands used on link l, $GB.size$ is the size of the guard band, and $rate.l$ is the transmission speed of link l.

$$BUGB = \frac{1}{hyper.T} \sum_{\forall l \in E} \frac{GB.cnt_l \times GB.size}{rate.l} \tag{4}$$

On the basis of the schedule obtained through schedule generation and compression, the schedule performance for the current chromosome can be measured by performing a fitness evaluation. For fitness, the E2E delay, jitter, and BUGB are used as performance indicators. However, these three performance indicators have different units and different semantics from each other. Therefore, it is necessary to normalize them before using them in the fitness function. For this, a large number of chromosomes are randomly generated, and their performance indicators are used as a basis for normalization. It turns out that these performance indicators follow a normal distribution. Thus, the performance indicators obtained from a chromosome could be converted into a z-score of the standard normal distribution using the mean and standard deviation of the randomly generated samples that followed a normal distribution. The performance indicators obtained in the normalization process were used as the parameters in the fitness function; they were expressed as $Fitness_D$, $Fitness_J$, and $Fitness_{GB}$. Finally, the final $Fitness_{total}$ used to evaluate the performance of the chromosome was defined as a weighted sum of the parameters as follows.

3. Reinforcement Learning Based Traffic Scheduling

In a TSN, finding the optimal schedule involves determining the transmission order of STs that maximizes the quality of service (QoS) of the network. Therefore, the TSN scheduling problem can be defined as a sequential decision-making problem. In this study, we define the scheduling problem as an MDP and apply reinforcement learning. First, the state is defined as the current arrangement of frames in a $1 \times N$ list, where N denotes the number of frames transmitted during the hyperperiod. An action is defined as the act of selecting a frame and placing it on a list. Figure 2 shows an example of the state space in which the scheduling problem is defined as an MDP. The initial state s_0 comprises N empty slots with no selected frames, and the state is determined by the selected frames and their order.

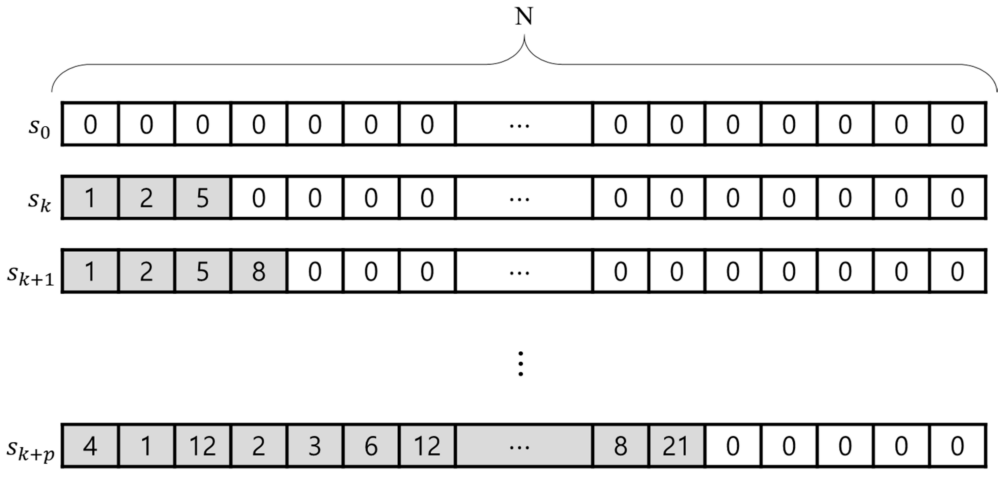


Figure 2. Example of state for schedule problem defined by the MDP.

Figure 3 illustrates the changes in the action space based on the action selection when the scheduling problem is defined as an MDP. Here, N represents the number of frames transmitted during the hyperperiod, as defined in the state space. The initial action space has a size corresponding to the arrangement of frames in a $1 \times N$ list. As actions are selected from the action space, their size decreases by one with each action, and the learning agent completes an episode by selecting N actions. This approach prevents the reselection of already placed frames, ensuring the completion of a full episode. For instance, in the case of a_0 in Figure 3, a maximum of two can be selected, and reducing the action space prevents exceeding the maximum selection count (where k denotes the flow type).

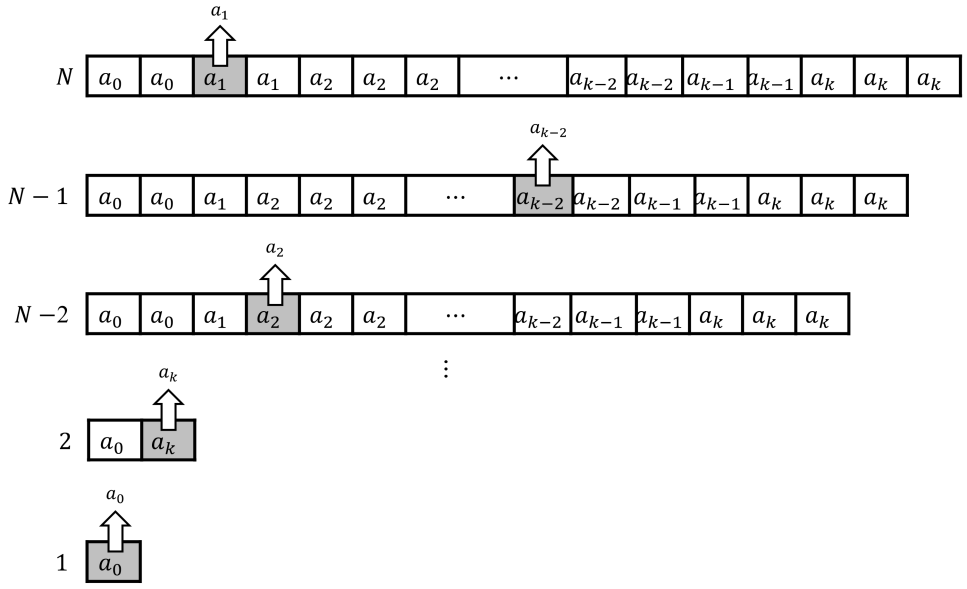


Figure 3. Example of action for schedule problem defined by the MDP.

Next, to calculate the reward, the total transmission start time of all the frames placed on the Gantt chart in the current state is expressed as $total_delay(s_t, a_t)$ in Equation (5). Here, n_ST represents the number of STs, and n_f represents the total number of f_i in the current state. L_m represents the link on which the m-th frame is transmitted, and n denotes the queue number at the egress port where the frame is transmitted. The reward function $r(s_t, a_t)$ is defined in Equation (6) as the transmission delay $f.D(a_t)$ of the selected frame divided by the sum of the total transmission start times of all frames calculated when taking action a_t in the current state s_t and the difference between the sum of the total transmission start times of all frames in the previous state s_{t-1} and $f.D(a_t)$. This allows us to determine the amount of delay before the transmission of the frame selected by the policy. If no delay occurs in the transmission link of the selected frame by taking action a_t, the reward has the highest value of one. If there is a delay in the link, the reward will be less than one. This method enables us to derive a policy that selects actions that result in the shortest delay.

$$total_{f.I}(s_t,\ a_t) = \sum_{i=1}^{n_ST} \sum_{m=1}^{n_f_i} f_{i,m}^{L_m,n}.\mathrm{I} \tag{5}$$

$$r(s_t,\ a_t) = \frac{f.D(a_t)}{total_{f.I}(s_t,\ a_t) - total_{f.I}(s_{t-1},\ a_{t-1}) + f.D(a_t)} \tag{6}$$

To verify whether the proposed reward function generates a schedule that minimizes delays, we compared the scheduling results based on the total accumulated rewards. It was observed that, as higher rewards were obtained, the network performance metric, E2E delay, was minimized.

4. Performance Evaluation

4.1. Evaluation Environment

To evaluate the performance of the schedule generated by the machine learning (ML)-based optimization algorithm, a network for the autonomous driving system of a large vehicle, such as a bus, was considered. As shown in Figure 4, the network was configured with 17 end stations and four SWs [26]. All links constituting the network were set to a 100 Mbps Ethernet full-duplex, and the information related to the 27 flows transmitted over the network was configured as summarized in Table 1. The experimental environment for performance evaluation is based on the autonomous hydrogen bus project conducted with H Company in Korea and is designed with a structure for autonomous driving, including four cameras, five radars, DCU, V2X, HVI, and ADR. Here, the IVN sensor refers to a module that converts CAN messages generated from vehicle parts into Ethernet messages and acts as a gateway for communication between sensors with different protocols.

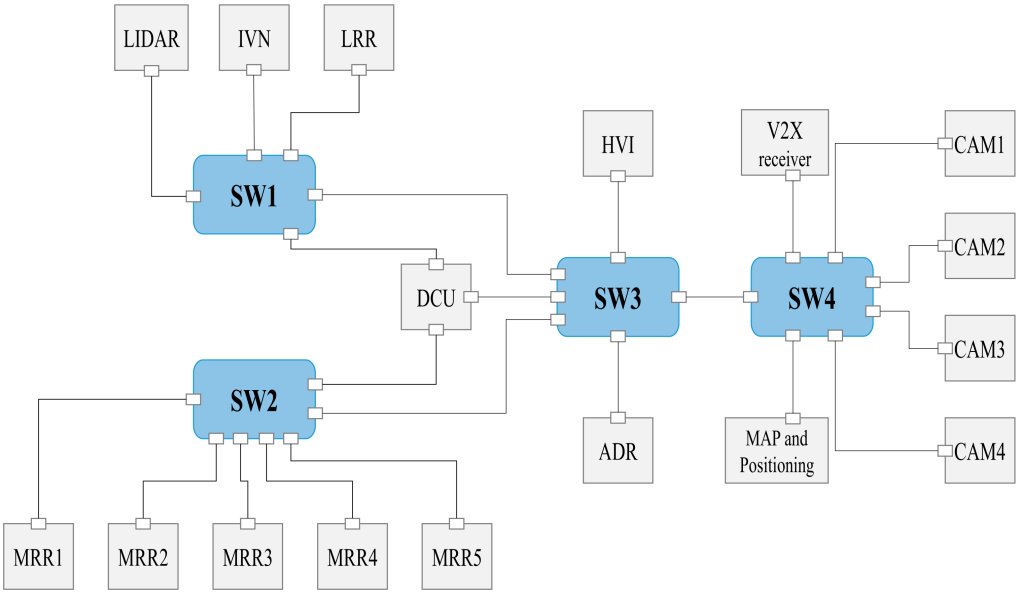


Figure 4. Network topology for performance evaluation of TSN optimal schedule algorithm.

All simulation models were programmed based on Python. DQN and A2C consumed 8 h and 5 h of training time, respectively, and the training model went through a lot of trial and error to avoid overfitting. The computer used for simulation training is an AMD 3600× processor, 32 GB of main memory, and a GeForce GTX 1080 ti graphics processing unit model. Here, the fitness was calculated as a weighted sum ($\alpha = 0.5$, $\beta = 0.3$, and $\gamma = 0.2$) of the performance indicators represented by z-scores. The lower the value, the better the performance. It was important to consider the most recent output of the sensors for autonomous driving; 0.5 was selected for α, which was the weight value of the E2E delay, and 0.3 was selected for β because sensor information had to be provided on a regular basis. Because the network traffic was not very large compared with the capacity of the network, γ, which was the weight of the loss because of the guard band, was set to a relatively low value, 0.2.

Table 1. Characteristics of flows on the network.

	Source	Destination	Period (ms)	Size (byte)	Transmission Delay (μs)	Route
1	LRR	DCU	50	125	10	[LRR, SW1], [SW1, DCU]
2	CAM1	DCU	20	750	60	[CAM1, SW4], [SW4, SW3], [SW3, DCU]
3	CAM2	DCU	20	750	60	[CAM2, SW4], [SW4, SW3], [SW3, DCU]
4	CAM3	DCU	20	750	60	[CAM3, SW4], [SW4, SW3], [SW3, DCU]
5	CAM4	DCU	20	750	60	[CAM4, SW4], [SW4, SW3], [SW3, DCU]
6	MRR1	DCU	50	125	10	[MRR1, SW2], [SW2, DCU]
7	MRR2	DCU	50	125	10	[MRR2, SW2], [SW2, DCU]
8	MRR3	DCU	50	125	10	[MRR3, SW2], [SW2, DCU]
9	MRR4	DCU	50	125	10	[MRR4, SW2], [SW2, DCU]
10	MRR5	DCU	50	125	10	[MRR5, SW2], [SW2, DCU]
11	LIDAR	IVN	20	363	29	[LIDAR, SW1], [SW1, IVN]
12	LIDAR	DCU	20	363	29	[LIDAR, SW1], [SW1, DCU]
13	LIDAR	ADR	20	363	29	[LIDAR, SW1], [SW1, SW3], [SW3, ADR]
14	MAP	DCU	100	625	50	[MAP, SW3], [SW3, DCU]
15	IVN	DCU	10	250	20	[IVN, SW1], [SW1, DCU]
16	IVN	LRR	10	250	20	[IVN, SW1], [SW1, LRR]
17	IVN	MRR1	10	250	20	[IVN, SW1], [SW1, SW3], [SW3, SW2], [SW2, MRR1]
18	IVN	MRR2	10	250	20	[IVN, SW1], [SW1, SW3], [SW3, SW2], [SW2, MRR2]
19	IVN	MRR3	10	250	20	[IVN, SW1], [SW1, SW3], [SW3, SW2], [SW2, MRR3]
20	IVN	MRR4	10	250	20	[IVN, SW1], [SW1, SW3], [SW3, SW2], [SW2, MRR4]
21	IVN	MRR5	10	250	20	[IVN, SW1], [SW1, SW3], [SW3, SW2], [SW2, MRR5]
22	DCU	IVN	10	250	20	[DCU, SW1], [SW1, IVN]
23	DCU	IVN	20	250	20	[DCU, SW1], [SW1, IVN]
24	V2X	MAP	20	250	20	[V2X, SW4], [SW4, MAP]
25	V2X	DCU	100	250	20	[V2X, SW4], [SW4, SW3], [SW3, DCU]
26	ADR	DCU	100	500	40	[ADR, SW3], [SW3, DCU]
27	HVI	DCU	100	125	10	[HVI, SW3], [SW3, DCU]

4.2. Deep Q Network (DQN)

The optimal solution for the TSN scheduling problem, defined as an MDP, was obtained using the DQN. The DQN first randomly selects actions and stores the resulting states, actions, and rewards as a dataset in the replay memory. The size of the replay memory was set to 750,000 entries. When the number of entries in the training dataset reached 720,000, the ε-greedy policy was applied. In the ε-greedy policy, probabilities of 1-ε to perform random exploration and ε to select the action with the highest value obtained from the neural network approximating the Q-function exist. The initial value of ε was set to 1 and was decreased by a factor of 0.9999 at each TD step to allow sufficient exploration at the beginning of the training. The minimum value of ε was set to 0.1 to maintain adequate exploration even after convergence.

A neural network approximating the Q-function was designed with an input layer receiving the state, two hidden layers, and an output layer providing the values of the actions. Each hidden layer comprised 512 nodes, and the ReLU function was used as the

activation function. A mini-batch of 512 data samples was randomly selected from the replay memory for training, and the learning rate was set to 0.01. The prediction network was updated at each TD step during training, whereas the target network was updated with the parameters of the prediction network at the end of each episode.

Figure 5 shows the total reward obtained in each episode over 5000 training iterations. With the application of the DQN, the total reward converged at approximately 5000 episodes. The observed variance around a specific value is a result of exploration owing to the ε-greedy policy. To evaluate the performance of the DQN-based schedule, the episode with the highest total reward among 5000 episodes was selected. The total reward for the selected episode was 295.55 points. Using the results of this episode, a schedule was generated, and three performance metrics were calculated. Table 2 lists the three performance metrics obtained from DQN and their normalized values. After the exploration phase for storing datasets in the replay memory, the highest total reward was achieved within approximately 2000 episodes. It took approximately 8 h to complete all 5000 episodes.

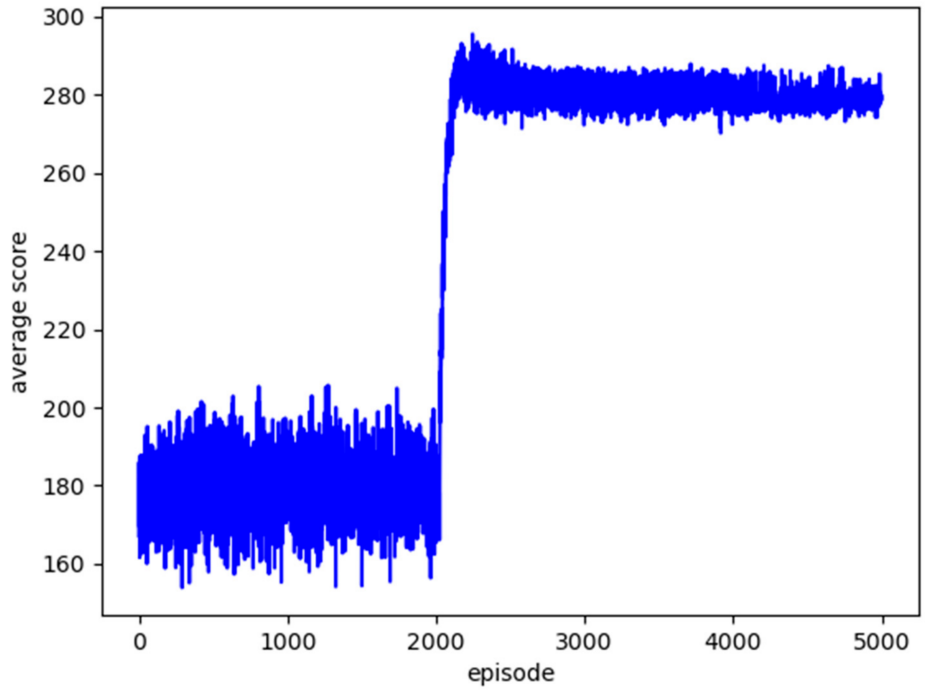


Figure 5. Total score per episode using DQN.

Table 2. DQN-based transmission scheduling result.

	E2E Delay ($Fitness_D$)	**Jitter** ($Fitness_J$)	**BUGB** ($Fitness_{GB}$)	$Fitness_{Total}$
DQN-based scheduling	6.1815×10^{-3} (-2.2842)	1.5293 (-2.8830)	9.3000×10^{-3} (-1.2000)	-2.2470

4.3. Advantage Actor-Critic (A2C)

The optimal solution for the TSN scheduling problem, defined as the MDP, was obtained using A2C. A2C requires two neural networks to approximate the policy and value functions. A policy network (actor) was constructed with an input layer receiving the state, a hidden layer with 128 nodes, and an output layer providing the probabilities of selecting each action. The hidden layer used the tanh activation function, whereas the output layer used the softmax activation function to ensure that the sum of the outputs was 1.

The value network (critic) was designed with an input layer that received the state, two hidden layers with 128 nodes each, along with an output layer that provided the value of the current state. The hidden layers also used the tanh activation function. The learning rate for training the neural network models was set to 0.01.

Figure 6 shows the total reward obtained in each episode over 5000 training iterations. With the application of A2C, an increase in the total reward was observed over approximately 5000 episodes, achieving a higher total reward than with Q-learning and DQN. To evaluate the performance of the schedule generated by A2C, the episode with the highest total reward among 5000 episodes was selected. The total reward for the selected episode was 310.17 points. Using the results of this episode, a schedule was generated, and three performance metrics were calculated. Table 3 lists the three performance metrics obtained from A2C and their normalized values. For A2C, the highest total reward was achieved after approximately 4000 episodes, taking approximately 4 h, while completing all 5000 episodes took approximately 5 h.

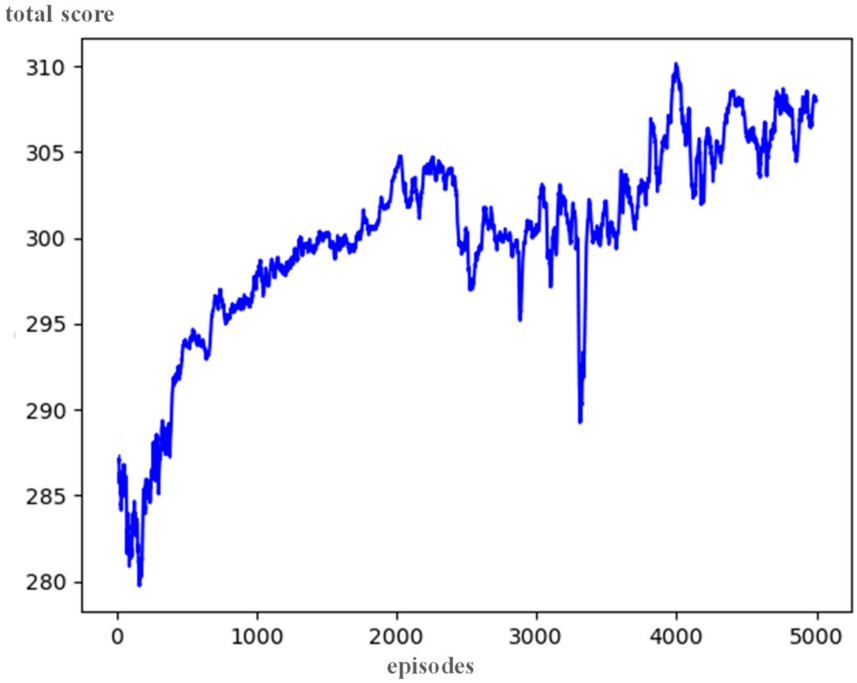


Figure 6. Total score per episode using A2C.

Table 3. A2C-based transmission scheduling results.

	E2E Delay ($Fitness_D$)	**Jitter** ($Fitness_J$)	**BUGB** ($Fitness_{GB}$)	$Fitness_{Total}$
A2C-based scheduling	3.9375×10^{-3} (−2.4054)	1.5074 (−2.8880)	9.2000×10^{-3} (−1.4000)	−2.3491

5. Conclusions

This study investigated the optimization of traffic schedules in TSN, a type of deterministic Ethernet, for autonomous vehicle networks. Specifically, the authors proposed network performance metrics and application methods for ML to optimize traffic schedules. This approach demonstrates that the NP-hard complexity of traffic scheduling can be optimized using ML algorithms. Furthermore, by comparing and analyzing the performance of each algorithm, suitable scheduling algorithms for the network requirements and development environments were identified.

Reinforcement learning algorithms such as DQN and A2C were used to optimize the traffic schedule. To evaluate the post-scheduling performance of the network, three

normalized performance metrics (E2E delay, jitter, and BUGB) and an evaluation function expressed as their weighted sum was proposed. The performance of each algorithm was evaluated using the topology of an actual autonomous vehicle network, and the strengths and weaknesses of each reinforcement learning method were compared and analyzed. Ultimately, we confirmed that artificial intelligence-based algorithms are effective in optimizing TSN traffic schedules.

Based on the results of time synchronization and schedule optimization for applying deterministic Ethernet to autonomous vehicle networks, the following conclusions were drawn:

First, in the TSN traffic scheduling problem, network performance is determined by the arrangement order of the frames that need to be scheduled. We confirmed the feasibility of applying artificial techniques, such as reinforcement learning, to determine the optimal solution for the scheduling problem.

Second, although the three selected performance metrics for schedule evaluation (E2E delay, jitter, and BUGB) have different meanings and units, they can be normalized using random sampling and represented as a standard normal distribution. Additionally, using the z-scores of the performance metrics, we confirmed that it was possible to evaluate the performance of the schedule by considering all three metrics through an evaluation function defined by their weighted sum.

Further theoretical and practical research is required to enhance the feasibility of applying deterministic Ethernet to autonomous vehicle networks. To further improve the optimization of TSN traffic schedules, scheduling must consider various variables and all delay factors that occur in real network environments instead of under simplified network conditions. Additionally, it is necessary to evaluate the performance in environments with traffic other than ST. Moreover, the impact of time synchronization and scheduling on vehicle control performance in autonomous networks needs to be assessed. Finally, this paper evaluated the applicability of the proposed method at the laboratory level, and if stability is secured in a real environment through additional research, the application will be shared.

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

Funding: This work was supported by the research grant of Gyeongsang National University in 2023.

Data Availability Statement: The datasets used and/or analyzed during the current study are available from the corresponding author upon reasonable request.

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

References

1. *J3016*; Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles. SAE: Pittsburgh, PA, USA, 2018.
2. Wang, S.; Zhou, H.; Zhao, H.; Wang, Y.; Cheng, A.; Wu, J. A Zero False Positive Rate of IDS Based on Swin Transformer for Hybrid Automotive In-Vehicle Networks. *Electronics* **2024**, *13*, 1317. [CrossRef]
3. Hank, P.; Suermann, T.; Müller, S. Automotive Ethernet, a holistic approach for a next generation in-vehicle networking standard. *Adv. Microsyst. Automot. Appl.* **2012**, *2012*, 79–89.
4. Park, C.S.; Park, S.K. Performance Evaluation of Zone-Based In-Vehicle Network Architecture for Autonomous Vehicles. *Sensors* **2023**, *23*, 669. [CrossRef] [PubMed]
5. Kim, M.H.; Lee, S.; Ha, K.N.; Lee, K.C. Implementation of a fuzzy-inference-based, low-speed, close-range collision warning system for the urban areas. *Proc. Inst. Mech. Eng. D* **2013**, *227*, 234–245. [CrossRef]
6. Ma, X.; Li, S.; Guan, Z.; Li, J.; Sun, H.; Wang, Y.; Guo, H. Time-Sensitive Networking Mechanism Aided by Multilevel Cyclic Queues in LEO Satellite Networks. *Electronics* **2023**, *12*, 1357. [CrossRef]
7. Guo, M.; Shou, G.; Liu, Y.; Hu, Y. Software-Defined Time-Sensitive Networking for Cross-Domain Deterministic Transmission. *Electronics* **2024**, *13*, 1246. [CrossRef]

8. Kwon, J.H.; Kim, M.H.; Kim, H.J.; Lee, M.W.; Lee, S. Driving Profile Optimization Using a Deep Q-Network to Enhance Electric Vehicle Battery Life. *J. Sens.* **2023**, *1*, 6684018. [CrossRef]
9. Sommer, J.; Gunreben, S.; Feller, F.; Kohn, M.; Mifdaoui, A.; Sass, D.; Scharf, J. Ethernet—A survey on its fields of application. *IEEE Commun. Surv. Tutor.* **2010**, *12*, 263–284. [CrossRef]
10. 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**, *107*, 1121–1131. [CrossRef]
11. IEEE 802.1Qbv—Bridges and Bridged Networks—Amendment 25: Enhancements for Scheduled Traffic. 2015. Available online: https://standards.ieee.org/ieee/802.1Qbv/6068/ (accessed on 7 July 2024).
12. IEEE 802.1Qbu—Bridges and Bridged Networks—Amendment 26: Enhancements for Frame Preemption. 2016. Available online: https://standards.ieee.org/ieee/802.1Qbu/5464/ (accessed on 7 July 2024).
13. Wan, T.; Ashwood-Smith, P. A performance study of CPRI over Ethernet with IEEE 802.1Qbu and 802.1Qbv enhancements. In Proceedings of the 2015 IEEE Global Communications Conference (GLOBECOM), San Diego, CA, USA, 6–10 December 2015; Volume 2015, pp. 1–6.
14. Smirnov, F.; Glaß, M.; Reimann, F.; Teich, J. Optimizing message routing and scheduling in automotive mixed-criticality time-triggered networks. In Proceedings of the 54th ACM/EDAC/IEEE Design Automation Conference, Austin, TX, USA, 18–22 June 2017; pp. 1–6. [CrossRef]
15. Gavriluţ, V.; Zhao, L.; Raagaard, M.L.; Pop, P. AVB-aware routing and scheduling of time-triggered traffic for TSN. *IEEE Access* **2018**, *6*, 75229–75243. [CrossRef]
16. Steiner, W. An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks. In Proceedings of the 31st IEEE Real-Time Systems Symposium, San Diego, CA, USA, 30 November–3 December 2010; pp. 375–384. [CrossRef]
17. Laursen, S.M.; Pop, P.; Steiner, W. Routing optimization of AVB streams in TSN networks. *SIGBED Rev.* **2016**, *13*, 43–48. [CrossRef]
18. 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 24th International Conference on Real-Time Networks and Systems, Brest, France, 19–21 October 2016; pp. 183–192. [CrossRef]
19. Craciunas, S.S.; Oliver, R.S. SMT-based task- and network-level static schedule generation for time-triggered networked systems. In Proceedings of the 22nd International Conference on Real-Time Networks and Systems, Versaille, France, 8 October 2014; pp. 45–54. [CrossRef]
20. Tămaş-Selicean, D.; Pop, P.; Steiner, W. Design optimization of TTEthernet-based distributed real-time systems. *Real-Time Syst.* **2015**, *51*, 1–35. [CrossRef]
21. Gavriluţ, V.; Pop, P. Scheduling in time sensitive networks (TSN) for mixed-criticality industrial applications. In Proceedings of the 14th IEEE International Workshop on Factory Communication Systems, Imperia, Italy, 13–15 June 2018; pp. 1–4. [CrossRef]
22. 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. [CrossRef]
23. 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]
24. Zheng, L.; Wei, G.; Zhang, K.; Chu, H. Traffic Classification and Packet Scheduling Strategy with Deadline Constraints for Input-Queued Switches in Time-Sensitive Networking. *Electronics* **2024**, *13*, 629. [CrossRef]
25. Huang, C.; Wang, Y.; Zhang, Y. Time-Sensitive Network Simulation for In-Vehicle Ethernet Using SARSA Algorithm. *World Electr. Veh. J.* **2024**, *15*, 21. [CrossRef]
26. Tindell, K.; Burns, A. Guaranteeing message latencies on controller area network (CAN). In Proceedings of the 1st International Can Conference, Geneva, Switzerland, 25–27 May 1994; pp. 1–11.

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.