The Time-Sensitive Networking Scheduling Algorithm Based on Q-learning

Jiayi Zhao
School of Computer Science and Engineering
Xi'an Technological University
Xi'an, 710021, China
E-mail: 1766614602@qq.com

Jing Cheng
School of Computer Science and Engineering
Xi'an Technological University
Xi'an, 710021, China
E-mail: chengjing@xatu.edu.cn

***Abstract*—Time-Sensitive Networking (TSN) occupies a vital position in modern communication domains, with the 802.1Qbv standard being an important network technology designed to meet real-time requirements. This standard requires network traffic to be transmitted within strict time windows, presenting challenges in network planning, necessitating efficient resource allocation and scheduling strategies. This paper addresses the 802.1Qbv planning problem through the introduction of reinforcement learning algorithms, offering an automated and intelligent solution. We have designed a reinforcement learning agent capable of perceiving network status, learning optimal resource allocation strategies, and dynamically adjusting in real-time environments. Through simulation and experimentation, we have validated the effectiveness of our proposed method, comparing it with traditional planning approaches. The contribution of this study lies in introducing a novel solution to the 802.1Qbv planning problem for time-sensitive networks, enhancing network resource utilization and performance. This approach offers strong support for the development and enhancement of TSN-like networks, holding significant importance for meeting the growing demands of real-time applications.**

Keywords-Time-Sensitive Networking; Reinforcement Learning; Network Planning; IEEE 802.1Qbv

I. INTRODUCTION

A. Research Motivation and Significance

Time-Sensitive Networking (TSN) plays a pivotal role in modern society, supporting a multitude of critical applications including industrial automation, intelligent transportation systems, real-time multimedia transmission in [7]. With the increasing demands for real-time performance, the planning and management of time-sensitive networks have become more complex and challenging. In particular, the introduction of the 802.1Qbv standard complicates network planning due to its requirement for network traffic to be transmitted within strict time windows, necessitating efficient resource allocation and scheduling strategies.

At present, the primary strategies for solving the 802.1Qbv scheduling issue include Satisfiability Modulo Theories (SMT) and linear programming. However, these methods have issues with planning time and adaptability to complex requirements.

This study introduces reinforcement learning as a means to enhance the adaptability of planning under complex conditions, optimize planning efficiency, and improve network performance.

B. Research Status at Home and Abroad

International scholars have extensively researched the planning and management of Time-Sensitive Networking (TSN). The scheduling issues of TSN were first influentially addressed in 2016, primarily using the Satisfiability Modulo Theories (SMT) method, which initially solved the planning problem. Reference [1] provides an introduction to TSN.

Farzaneh and colleagues developed an automated scheduling synthesis tool for supporting TSN using graphical modeling in [2]. They also optimized the solution algorithms for planning based on previous studies. Reference [3] use array theory encoding to solve the 802.1Qbv problem. M. Pahlevan and others proposed a heuristic scheduling algorithm based on genetic algorithms in [4]. Gavriluţ and colleagues introduced a scheduling algorithm based on Greedy Randomized Adaptive Search Procedures (GRASP) in [5]. In 2019, the TSNSCHED tool was introduced in [6]. It takes topology as input, utilizes SMT for solving, and supports stream planning for TSN unicast and multicast. Li C and colleagues proposed an integer linear programming based joint routing and scheduling method for multicast time-sensitive traffic in [13]. They reduced the scale of the scheduling problem by pruning the topology and grouping the traffic.

Mai T L and others proposed the use of machine learning methods in [8], including supervised and unsupervised learning, to explore the solution space of well-constructed TSN. Zhong C and colleagues were the first to propose using deep reinforcement learning to address the dynamic scheduling problem of TT streams in [9]. Their solution can promptly recover and reschedule the affected TT streams when network topology changes occur. Jia H and others proposed a deep reinforcement learning scheduling framework in [10], for incremental scheduling of TT streams. Based on this, they designed a three-step selection paradigm to mitigate the issue of a vast action search space. In [11], Jin X and colleagues proposed a method for scheduling large-scale data under the condition of limited Schedule Entries.

In 2022, Daniel B and others introduced a heuristic scheduling algorithm called HERMES in [12], which utilizes multiple scheduling queues to enhance the schedule ability of time-sensitive traffic.

Overall, researchers both internationally and domestically have recognized the importance of TSN and have achieved a series of results using different research methods and technologies. However, due to the complexity and real-time demands of TSN, this field remains challenging and requires further in-depth study and innovation. This research aims to contribute to the study of TSN planning issues in the domestic context by introducing reinforcement learning methods to improve network performance and resource utilization.

II. Design of Algorithm

Network Mathematical Model

Time-Sensitive Networking is a network architecture made up of end systems (ES), switches (SW), and full-duplex physical links. This paper abstracts the time-sensitive network as a directed graph $G = (N, L)$.

N symbolizes all device nodes within TSN, encompassing both switches and end systems. End systems are responsible for sending and receiving traffic that carries data within the network. Switches comprise multiple input and output ports and are responsible for forwarding frames from input ports to the corresponding output ports based on the destination node.

L denotes the set of physical connections within TSN, with each element signifying a unidirectional, simplex physical link. These links are the physical medium connecting the output ports of network nodes and are responsible for the actual transmission of traffic.

Fig. 1 illustrates a basic model of TSN topology., including 2 switches and 4 end systems, interconnected by a total of 10 unidirectional links.

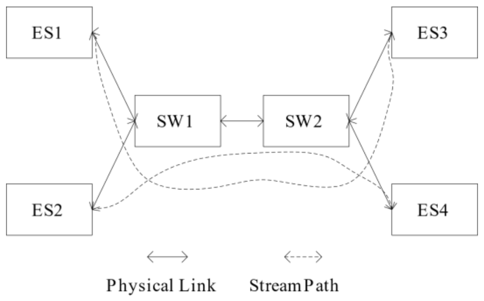

Figure 1. Simple TSN Network Model

In the model, a physical link is represented as a unidirectional link $[n_a, n_b]$, defined by a triplet $\langle [n_a, n_b].speed, [n_a, n_b].count, [n_a, n_b].delay \rangle$. Here, $[n_a, n_b].speed$ denotes the transmission rate, $[n_a, n_b].count$ indicates the number of output port queues connected to node n_a on the link, and $[n_a, n_b].delay$ represents the propagation delay.

The traffic model of TSN mainly includes Scheduled Traffic (ST) streams, Audio Video

Bridging (AVB) streams, and Best Effort (BE) streams in [15], with the priority decreasing in that order. The scheduling only considers the ST stream problem. The entirety of ST streams within the network is represented by the symbol S. For an ST stream $s_i \in S$ originating from node n_1 and destined for node n_{n_i}, passing through intermediate nodes $n_2, \ldots, n_{n_i - 1}$, the transmission path of s_i is represented as $s_i = \left\{ [n_1, n_2], \ldots, [n_{n_i-1}, n_{n_i}] \right\}$. The flow instance of s_i on the link $[n_a, n_b]$ is represented as $s_i^{[n_a, n_b]}$, and each s_i is defined by a triplet $\langle s_i.\text{len}, s_i.\text{period}, s_i.\text{e2e} \rangle$. Here, $s_i.\text{len}$ indicates the size of stream s_i, the quantity of data transmitted within each period; $s_i.\text{period}$ denotes the period length of stream s_i; $s_i.\text{e2e}$ represents the upper limit of tolerable end-to-end delay(ED) for a stream s_i.

The hyper period of the link $[n_a, n_b]$ is defined as the least common multiple of all scheduled stream's periods passing through the link. The scheduling algorithm is only required to organize the streams within a single hyper period and repeat this schedule across multiple hyper periods.

All traffic is transmitted in units of frames in [14]. $F_i^{[n_a, n_b]}$ denotes all frame instances of stream s_i transmitted on the link $[n_a, n_b]$, where $f_{i,k}^{[n_a, n_b]}$ represents the kth frame instance in the set. Each frame instance is defined by a triplet $\left\langle f_{i,k}^{[n_a, n_b]}.offset, f_{i,k}^{[n_a, n_b]}.trans\_dur, f_{i,k}^{[n_a, n_b]}.period \right\rangle$. Here, $f_{i,k}^{[n_a, n_b]}.offset$ represents the offset of the instance's transmission time relative to the start of the hyper period, satisfying Equation (1):

$$f_{i,k}^{[n_a, n_b]}.offset \in \left[0, f_{i,k}^{[n_a, n_b]}.\text{period} \right] \qquad (1)$$

$f_{i,k}^{[n_a, n_b]}.trans\_dur$ represents the transmission delay of the instance, determined by its size and the link's bandwidth. $f_{i,k}^{[n_a, n_b]}.\text{period}$ represents the duration of the instance, equivalent to the stream's cycle.

Scheduling Constraints

Scheduling constraints are essential conditions for determining the correctness and effectiveness of scheduling results. The traffic scheduling problem in TSN can be defined as an optimization problem: assigning appropriate transmission slots to all planned streams in the network, with the objective of maximizing optimization indicators while satisfying all scheduling constraints. The specific constraints are as follows:

Frame Periodicity Constraint: Planned streams are periodic, and it must be ensured that each frame completes transmission before the end of its period. This can be expressed as for $\forall s_i \in S, \forall [n_a, n_b] \in s_i, \forall f_{i,k}^{[n_a, n_b]} \in F_i^{[n_a, n_b]}$, satisfying Equation (2):

$$\begin{cases} f_{i,k}^{[n_a, n_b]}.offset \geq 0, \\ f_{i,k}^{[n_a, n_b]}.offset \leq \\ f_{i,k}^{[n_a, n_b]}.period - f_{i,k}^{[n_a, n_b]}.trans\_dur. \end{cases} \qquad (2)$$

Link Conflict-Free Constraint: To ensure the isolation and correctness of the flow's transmission, it is necessary to guarantee that each frame exclusively occupies the queue of the output port and the corresponding physical link during the same transmission slot. This can be represented by Equation (3):

$$\begin{aligned} & \forall [n_a, n_b] \in L \ \forall f_{i,k}^{[n_a, n_b]} \in F_i^{[n_a, n_b]}, \\ & f_{j,l}^{[n_a, n_b]} \in F_j^{[n_a, n_b]}, i \neq j : \\ & f_{i,k}^{[n_a, n_b]}.offset + \alpha \times f_{i,k}^{[n_a, n_b]}.period \geq \\ & f_{j,l}^{[n_a, n_b]}.offset + \beta \times f_{j,l}^{[n_a, n_b]}.period + \\ & f_{j,l}^{[n_a, n_b]}.trans\_dur \vee \\ & f_{j,l}^{[n_a, n_b]}.offset + \beta \times f_{j,l}^{[n_a, n_b]}.period \geq \\ & f_{i,k}^{[n_a, n_b]}.offset + \alpha \times f_{i,k}^{[n_a, n_b]}.period + \\ & f_{i,k}^{[n_a, n_b]}.trans\_dur. \end{aligned} \qquad (3)$$

Frame Transmission Constraint: The transmission start time for a given frame on the subsequent link must not precede the completion time of its transmission on the prior link. This can be represented by Equation (4):

$$\begin{aligned}&\forall s_i \in S, [n_a, n_x], [n_x, n_b] \in s_i\\&f_{i,k}^{[n_a,n_x]} \in F_i^{[n_a,n_x]}, f_{i,k}^{[n_x,n_b]} \in F_i^{[n_x,n_b]}:\\&f_{i,k}^{[n_a,n_x]}.offset + f_{i,k}^{[n_a,n_x]}.trans\_dur +\\&[n_a,n_x]\,delay + \delta \le f_{i,k}^{[n_x,n_b]}.offset\end{aligned} \tag{4}$$

Flow Isolation Constraint: To prevent the interleaving of frame order in the scheduling queue, restrictions are imposed on any two Scheduled Traffic (ST) streams arriving at the same switch. If a frame from one ST stream has already arrived at the switch, frames from another ST stream cannot reach the same output port until all frames of the first stream have been completely sent to the output port. This can be represented by Equation (5):

$$\begin{aligned}&\forall [n_a,n_b] \in L\ s_i^{[n_a,n_b]}, s_j^{[n_a,n_b]} \in S, i \ne j:\\&f_{i,ni}^{[n_a,n_b]}.offset + \alpha \times s_i.period + \delta \le\\&f_{j,1}^{[n_y,n_a]}.offset + \beta \times s_j.period +\\&[n_y,n_a]\,delay \vee f_{j,nj}^{[n_a,n_b]}.offset +\\&\beta \times s_j.period + \delta \le f_{i,1}^{[n_x,n_a]}.offset +\\&\alpha \times s_i.period + [n_x,n_a]\,delay.\end{aligned} \tag{5}$$

End-to-End Delay Constraint: The total delay experienced by each stream from source to destination must not exceed its specified maximum allowable delay. This can be represented by Equation (6):

$$\begin{aligned}&\forall s_i \in S: f_{i,ni}^{[n_y,n_b]}.offset +\\&f_{i,ni}^{[n_y,n_b]}.trans\_dur -\\&f_{i,1}^{[n_a,n_x]}.offset \le s_i.e2e\end{aligned} \tag{6}$$

Scheduling Optimization Indicators

In the TSN scheduling process, it is first necessary to satisfy the basic scheduling constraints to ensure the correctness of the scheduling results. Subsequently, the aim is to provide low-latency transmission services and enhance network transmission performance to achieve higher quality scheduling results. Therefore, two optimization indicators will be set to improve the quality of scheduling from the perspectives of slot utilization balance and ED.

A classic optimization goal in a TSN scheduling problem is the minimization of ED. To measure the quality of scheduling results, this paper establishes an optimization indicator for the ED of ST streams, aimed at achieving scheduling results that meet the scheduling constraints while minimizing the ED of ST streams. The ED denotes the duration from when the initial frame of the traffic begins transmission at the source node to the completion of the last frame's transmission. In conjunction with the TSN scheduling model, the calculation of ED for stream s_i is as Equation (7):

$$\begin{aligned}e2e_i = {}& f_{i,ni}^{[n_{i(n_i-1)},n_{n_i}]}.offset +\\& f_{i,ni}^{[n_{n_i-1},n_{n_i}]}.trans\_dur - f_{i,1}^{[n_{i1},n_{i2}]}.offset\end{aligned} \tag{7}$$

Define the ED delay vector for s_i as ED_i, which can be expressed as Equation (8):

$$ED_i = 1 - \frac{e2e_i}{s_i.e2e} \tag{8}$$

It can be concluded that the smaller $e2e_i$ is, the larger ED_i will be, indicating better network performance. This paper calculates the overall ED of all ST streams using the Average End-to-end Delay (AED) method. This approach is used to measure the network's end-to-end delay indicator:

$$AED = \sum_{i=1}^{N} \frac{ED_i}{N} \tag{9}$$

Another optimization goal is the uniformity of time slots. More uniform time slots can reduce the average delay of messages other than Scheduled Traffic (ST) messages. As long as the delay of ST messages is kept within the required limits, this is acceptable. The planned time slots are shown in Fig.2 as follows:

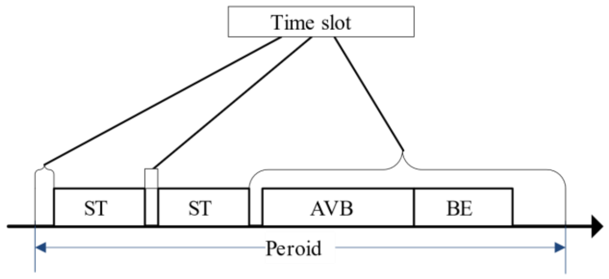

Figure 2. TSN time slot

The network's slot balance indicator primarily assesses the balance of idle slots on each link. The balance of each link is abstracted into a Link Balance Vector (LB), which is described using the standard deviation method. For link $[n_a, n_b]$, the calculation of the Link Balance Vector (LB) is as per Equation (10):

$$LB_{[n_a,n_b]} = 1 - \sqrt{\frac{\sum_{i=1}^{n^{[n_a,n_b]}} \left(slot_i^{[n_a,n_b]} - \overline{slot} \right)^2}{n^{[n_a,n_b]}}} \quad (10)$$

In this context, $n^{[n_a,n_b]}$ represents the number of idle slots on the $[n_a, n_b]$ link. $slot_i^{[n_a,n_b]}$ denotes the length of the i-th idle slot on $[n_a, n_b]$. $\overline{slot}$ is the average length of idle slots on the $[n_a, n_b]$link.

The overall Network Link Balance (NLB) for the network can be described by the average of all Link Balance Vectors across the links, as shown in Equation (11):

$$NLB = \sum_{i=1}^{N} \frac{LB_i}{N} \quad (11)$$

N signifies the total count of links within the network.

Design of the Scheduling Algorithm

The Algorithm introduces the concept of time discretization, converting the continuous time interval into a set of transmission moments. This transforms the problem of choosing transmission moments for traffic into a multi-treasure hunt in a three-dimensional space, with each planned stream acting as a searcher. The global link slot allocation in the network is designed as the environment. The scheduling of traffic at each hop along its transmission path is designed as the state space, where the current state fully characterizes the process, conforming to Markov properties. The action of choosing a transmission moment for the traffic on that link in the current state is designed as the action space, where the action space is the set of transmission moments obtained by discretizing the traffic's period. Regarding exploration strategy, this paper improves upon the ε-greedy strategy, adaptively adjusting the exploration probability ε based on iteration count, allowing the scheduling results to better converge to the optimal state. Combined with the optimization indicators mentioned in the previous section, a reward function is designed to evaluate scheduling actions and update the Q-table, until the algorithm reaches the maximum number of iterations, thus obtaining the final scheduling results.

The task of the TSN scheduling algorithm is to calculate the transmission moments for frames of all planned streams on the transmission links. The TSN scheduling problem has Markov properties; by continuously applying a one-hop scheduling policy to the network, observing changes in the state of the network environment, and obtaining immediate reward values, the algorithm trains an adaptive traffic scheduling model. After numerous iterations of training, an optimal behavior strategy is obtained, guiding the agent to execute the optimal scheduling actions. The process of the algorithm is as follows:

1) Parse configuration files, initialize network topology information, traffic collection, etc.

2) Initialize the scheduling model, including the environment, Q-table collection, learning rate α, discount factor γ, maximum number of iterations (episodes), exploration probability ε, state space, and action space.

3) Use an adaptive exploration strategy to select transmission moments, calculate the ε exploration factor, randomly select from the action space with a probability of ε, and select the transmission moment with the highest Q value from the Q-table with a probability of 1-ε.

4) Execute scheduling actions, move to a new state, and check if scheduling constraints are met. If not, provide negative feedback; if constraints are met, mark in the environment.

5) Update the Q-table according to the action record, regardless of whether the scheduling is successful or not.

6) If all ST streams are scheduled, indicating a successful scheduling, calculate the ED and NLB indicators of the scheduling results, put the weighted R value into the experience pool, and give positive reward feedback according to the reward function, indicating the end of this iteration round, then proceed to Step 7. If not all scheduling is completed and still in an intermediate state, go back to Step 3 and proceed with the scheduling process.

7) Ascertain if the count of iterations (episodes) has hit its upper limit. If not, go back to Step 3 and continue iterative training until the maximum number of iterations is reached, concluding the training.

After the algorithm training is completed, the final Q-table collection is obtained. The Q value indicates the superiority of choosing different transmission moments for traffic in each state; the larger the Q value of a transmission moment, the closer it can approach the scheduling goal. According to the final Q-table, the optimal transmission slots are planned for each planned stream on each link, serving as the basis for generating the gate control list.

The environment (Env) represents the overall slot allocation situation of the links in the current network, which can be seen as the mapping information between the traffic and the transmission moments on the transmission link. The definition of Env is shown in Equation (12):

$$Env = \left\{ s_i \rightarrow \left[n_a, n_b \right] \rightarrow f_{i,k}^{\left[n_a, n_b \right]}.offset \right\} \quad (12)$$

The scheduling space is divided into three types of states: effective, intermediate, and failure. A successful scheduling state is represented by all planned streams having generated scheduling schemes that meet the scheduling constraints. An intermediate state occurs when all streams scheduled so far meet the constraints, while a failure state arises when the current scheduling includes scenarios that do not meet the constraints. As shown in Fig.3 below, the diagram depicts the scheduling of three streams, with the top showing a successful state, the middle an intermediate state, and the bottom displaying a failed state due to a conflict.

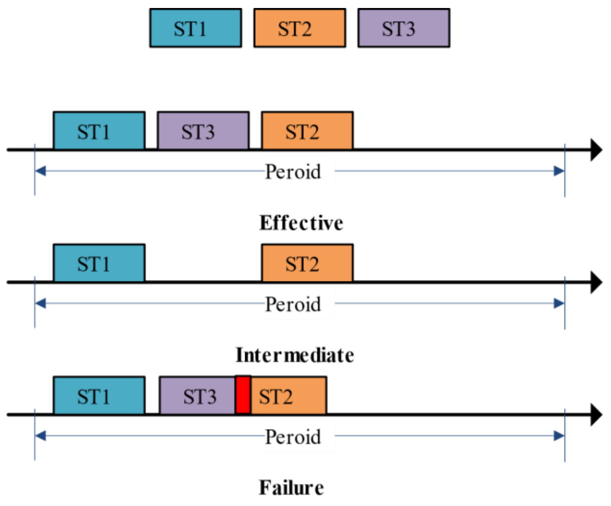

Figure 3. TSN Scheduling space status example

Let *State* represent the state space, where $State_t \in State$ denotes the scheduling state at moment t. The definition of $State_t$ is as shown in Equation (13):

$$State_t = \left\langle S, L, Env \right\rangle \quad (13)$$

The action space is a set of time points obtained by discretizing the size of the traffic cycle. Let A represent the action space, and $a_t \in A$ denote the action of the agent at moment t, which is the transmission time chosen for the traffic. Thus, the definition of a_t is as per Equation (14):

$$a_t = sche\left\langle s_i, \left[n_a, n_b \right] \ f_{i,k}^{\left[n_a, n_b \right]}.offset \right\rangle \quad (14)$$

Where $sche$ — on the $[n_a, n_b]$ link, the allocated transmission time for the frame instance of stream s_i is $f_{i,k}^{[n_a,n_b]}.offset$.

During the scheduling process in TSN, agents need to continuously select scheduling actions from

the action space for learning. This paper adopts an improved ε-greedy strategy for action selection, exploring with a probability of ε, and exploiting with a probability of 1-ε.

The calculation method for the exploration probability ε is as per Equation (15):

$$\varepsilon = \cos\left(\frac{\pi}{2} * \frac{cur\_epi}{episodes}\right) \quad (15)$$

In the learning process of TSN scheduling, each time a scheduling action is applied to the environment, a new scheduling state is generated. The reward function can provide a corresponding feedback evaluation for the new scheduling state.

When the new state s_{i+1} does not meet the scheduling constraints, it indicates that s_{i+1} cannot ensure that all planned streams in the current network can be correctly scheduled, and is considered a failure scheduling state. In this case, the reward function returns a negative reward value. When the new state s_{i+1} meets the scheduling constraints, and all ST streams in the network have been scheduled, a positive reward value is returned. When the new scheduling state s_{i+1} meets the scheduling constraints, but there are still unscheduled planned streams in the network, this scheduling state is an intermediate state, and a reward value of 0 is returned. In summary, the definition of the reward is as shown in Equation (16):

$$R(s_i, a_i) = \begin{cases} -1, \ failure \\ 0, \ intermediate \\ \lambda_1 * (\exp\left(1 - \frac{e2e}{CC}\right) - 1) + \\ \lambda_2 * (\exp\left(1 - \frac{NLB}{NLB_{MAX}}\right) - 1) \\ , \ effective \end{cases} \quad (16)$$

In the Q-learning-based TSN scheduling algorithm, during the process of generating scheduling results, it is necessary to design a Q-table to store reward values. After executing a scheduling action a_i in any state s_i, the reward function returns a reward value, denoted as $Q(s_i, a_i)$. In the scheduling algorithm designed in this paper, each planned stream corresponds to a Q-table. Each Q-table is an n*m experience matrix, where n is the number of links that the traffic route passes through, and each row of the Q-table, from top to bottom, corresponds to each transmission link of the traffic in sequence; m is the number of time points obtained by discretizing the period of the traffic. The Q-table is initially set to all zeros and is continuously updated with values through learning.

III. RESULTS

In this paper, experiments will be conducted based on the optimization indicators mentioned earlier, to compare and analyze the Q-learning-based TSN scheduling algorithm with the SMT scheduling algorithm, thereby verifying the advantages and disadvantages of the scheduling algorithms.

The subject of the TSN scheduling algorithm is the planned streams. The experimental data mainly includes the collection of planned streams awaiting scheduling in the network and the topology information. The TSN topology used in the experiment is illustrated in Fig.4 below. This network topology comprises 16 network end systems, 8 TSN network switches, and 31 full-duplex physical links. The assumption is made that every network device and physical connection within the network shares the same model, with identical bandwidth, and the lengths of the links are equal.

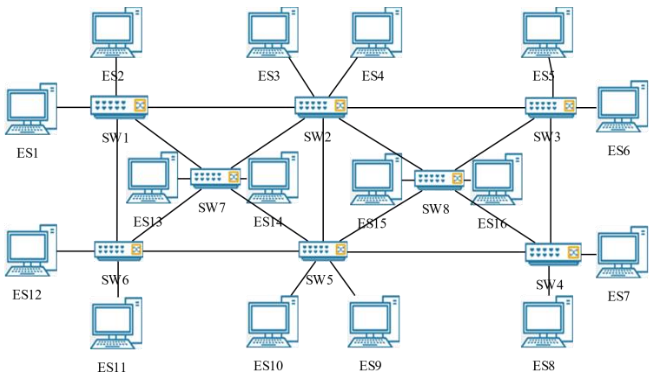

Figure 4. The experimental network topology

In the course of the experiment, the size, period, source, and destination of the ST streams are generated in random. The period of ST streams is chosen at random from the options of 1ms, 2ms, 4ms, 8ms, and 16ms. If the size of an ST stream exceeds the MTU, it can be divided into a collection of frames. According to standard Ethernet limitations, each frame is between 64B and 1518B. Given that the bandwidth of all network links is configured to 1000Mbps, the transmission delay for a frame at the output port varies between 5.12 microseconds and 121.44 microseconds.

The experiment injected 80 planned streams into the network for scheduling, with the size, period, source, and destination systems of each stream randomly generated. Assuming that the transmission paths of the flows have been determined by a specific routing algorithm, there are a total of 302 hops, and the hyper period for scheduling is 4000 us. The parameters for the scheduling algorithm are configured with a maximum of 50,000 iterations, a learning rate of 0.6, a discount factor of 0.9, an end-to-end delay index weight λ_1 of 0.5, and a NLB weight λ_2 of 0.5.

After obtaining the experimental results and analyzing the data, we listed the maximum and minimum values of the NLB for the scheduling results during the 50,000 iterations of training for these two scheduling algorithms, as shown in Fig.5:

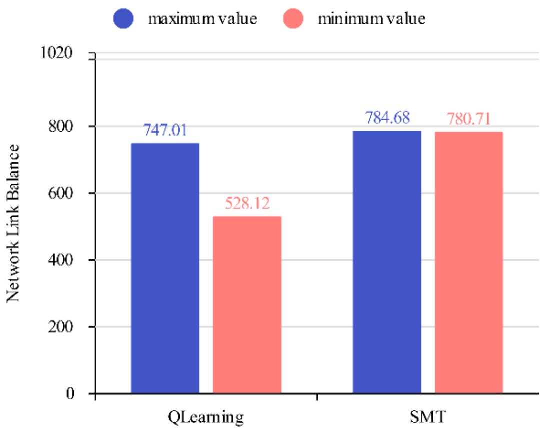

Figure 5. Comparison of NLB

As depicted in Fig.6 below, the change curve of the optimal value of the NLB of the scheduling results is given for these two algorithms under successful scheduling conditions across 50,000 iterations of training. In the graph, the horizontal axis denotes the algorithm's iteration count, whereas the vertical axis reflects the NLB value of the scheduling results. It can be observed that NLB of the scheduling results obtained by the SMT scheduling algorithm remains unchanged during the iterations. The reason is that the SMT solver lacks exploration capability, and the solution results are only related to the solution space of the scheduling problem. The NLB of the Q-learning scheduling algorithm shows a significant decline. During the training process, the algorithm continuously adjusts the scheduling strategy using an adaptive exploration strategy and gradually converges to higher-quality scheduling results under the guidance of the reward function.

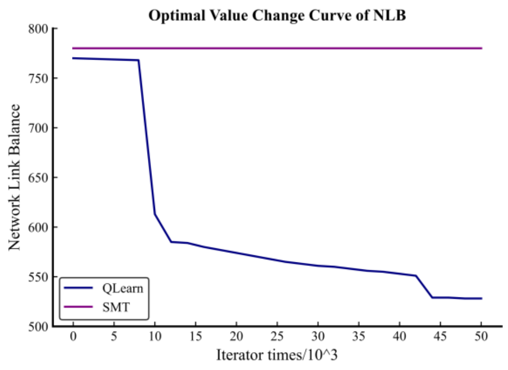

Figure 6. Optimal Value Change Curve of NLB

It can be deduced that the Q-learning based TSN network scheduling algorithm considerably surpasses the SMT algorithm in achieving a higher NLB. Moreover, the Q-learning-based approach, as compared to the SMT algorithm, possesses the capability to continuously optimize based on experience during iterations, converging towards higher-quality scheduling results in terms of the index.

IV. Conclusion

This paper revolves around the planning problem of TSN, involving mathematical analysis and modeling, and defines optimization indicators. Subsequently, a TSN planning and scheduling method based on Q-learning is proposed. Employing a discretization approach, the scheduling problem parameters and functions in Q-learning are defined and trained. Finally, experiments are conducted to compare the results of this method with traditional approaches. It is evident that the TSN scheduling method based on Q-learning shows significant superiority over traditional algorithms in terms of indicators such as NLB.

This paper proposes a TSN scheduling method based on Q-learning, which also provides insights for subsequent TSN planning problems. For example, it raises questions such as whether there are faster training-based methods that can be applied to TSN planning issues, or whether it's possible to combine current popular approaches, such as large models, for planning purposes.

References

[1] Messenger J L, Time-Sensitive Networking: An Introduction [J]. IEEE Communications Standards Magazine, 2018, 2(2): 2933. DOI:10.1109/MCOMSTD. 2018.1700047.

[2] Farzaneh M H, Kugele S, Knoll A. A graphical modeling tool supporting automated schedule synthesis for time-sensitive networking [C]. In: 2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2017: 1-8.

[3] Oliver R S, Craciunas S S, Steiner W. IEEE 802.1Qbv Gate Control List Synthesis using Array Theory Encoding [C]//IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS). IEEE, 2018. DOI:10.1109/RTAS.2018.00008.

[4] Pahlevan M, Obermaisser R. Genetic Algorithm for Scheduling Time-Triggered Traffic in Time-Sensitive Networks[C]//2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation. IEEE, 2018. DOI:10.1109/ETFA.2018.8502515.

[5] Gavrilut V, Pop P. Scheduling in time sensitive networks (TSN) for mixed-criticality industrial applications [C]//14 Th IEEE International Workshop on Factory Communication Systems. IEEE, 2018:1-4. DOI:10.1109/WFCS.2018.8402374.

[6] Santos A C T D, Schneider B, Nigam V, TSNSCHED: Automated Schedule Generation for Time Sensitive Networking[C]//2019 Formal Methods in Computer Aided Design (FMCAD). 2019. DOI:10.23919/FMCAD.2019.8894249.

[7] Bello L L, Steiner W. A perspective on IEEE time-sensitive networking for industrial communication and automation systems [J]. Proceedings of the IEEE, 2019, 107(6): 1094-1120.

[8] Mai T L, Navet N, Migge J. A hybrid machine learning and schedulability analysis method for the verification of TSN networks [C]. In: 2019 15th IEEE International Workshop on Factory Communication Systems (WFCS), 2019: 1-8.

[9] Zhong C, Jia H, Wan H, et al. DRLS: A Deep Reinforcement Learning Based Scheduler for Time-Triggered Ethernet[C]//2021 International Conference on Computer Communications and Networks (ICCCN). IEEE, 2021: 1-11.

[10] Jia H, Jiang Y, Zhong C, et al. TTDeep: Time-Triggered Scheduling for Real-Time Ethernet via Deep Reinforcement Learning [C]. In: 2021 IEEE Global Communications Conference (GLOBECOM), 2021: 1-6.

[11] Jin X, Xia C, Guan N. Real-time Scheduling of Massive Data in Time Sensitive Networks with a Limited Number of Schedule Entries [J]. IEEE Access, 2020, PP (99): 1-1. DOI:10.1109/ACCESS.2020.2964690.

[12] Bujosa D, Ashjaei M, Papadopoulos A V, et al. HERMES: Heuristic multi-queue scheduler for TSN time-triggered traffic with zero reception jitter capabilities [C]. In: Proceedings of the 30th International Conference on Real-Time Networks and Systems, 2022: 70-80.

[13] Li C, Zhang C, Zheng W. Joint Routing and Scheduling for Dynamic Applications in Multicast Time-Sensitive Networks [C]//IEEE International Conference on Communications Workshops. IEEE, 2021. DOI:10.1109/ICCWorkshops50388.2021.9473540.

[14] Nasrallah A, Thyagaturu A S, Alharbi Z, et al. Ultra-low latency (ULL) networks: The IEEE TSN and IETF DetNet standards and related 5G ULL research [J]. IEEE Communications Surveys & Tutorials, 2018, 21(1): 88-145.

[15] Reusch N, Zhao L, Craciunas S S, et al. Window-Based Schedule Synthesis for Industrial IEEE 802.1Qbv TSN Networks [C]// 2020 16th IEEE International Conference on Factory Communication Systems (WFCS). IEEE, 2020.