ISSN: 2414-1895 DOI: 10.6919/ICJE.202401_10(1).0019

Research on Communication Transmission Technology for Delay Sensitive Services

Yan Liu*, Runhua Yang, Huifang Liu

State Grid Henan Information & Telecommunication company, China

*519271097@qq.com

Abstract

With the construction of new power systems, a large number of new businesses such as distributed new energy and flexible loads have emerged. Some of these businesses are highly sensitive to delays, requiring delays to be in the range of seconds or even milliseconds, and are referred to as delay-sensitive businesses. Some business scenarios in power production, such as real-time monitoring and unmanned inspection, are highly sensitive to delays. If the delay cannot be controlled within a limited range, it will inevitably affect the stability of the power system and reduce the efficiency of power production. This paper proposes a delay-sensitive traffic scheduling mechanism based on circular queues and forwarding to address the inflexible traffic scheduling issues caused by the simple use of flow sorting, offset search, and resource judgment in traditional circular queue and forwarding (CQF), thereby improving the scheduling performance of delay-sensitive businesses.

Keywords

Delay-sensitive; Monitoring; Unmanned Inspection.

1. Scheduling Architecture based on CQF

The scheduling architecture based on CQF is shown in Fig. 1. TSN consists of a data plane and a control plane [2]. The data plane consists of physical network elements, including TSN switches and hosts. It can provide real-time data flow and network topology information from the industrial site to the control plane. The control plane stores global information and plans network traffic scheduling based on the information provided by the data plane. The control plane consists of the following three parts:

•Model: The topology and flow models provide mathematical descriptions and abstractions for the network topology and data flow. The switch model specifies the forwarding of data frames in CQF. The generated scheduling plan must follow the principles in the switch model. A detailed description of these models is provided in the Chapter 3.

•Scheduling algorithm: Used to generate scheduling plans and to issue them to hosts and switches in the data plane. This algorithm is detailed in Chapter 5.

•Constraints and optimization objectives: Constraints restrict the transmission of data flows in the network. The optimization objectives are the goals of the scheduling algorithm. The scheduling algorithm must satisfy the constraints and achieve the optimization objectives.

Volume 10 Issue 1, 2024
DOI: 10.6919/ICJE.202401_10(1).0019

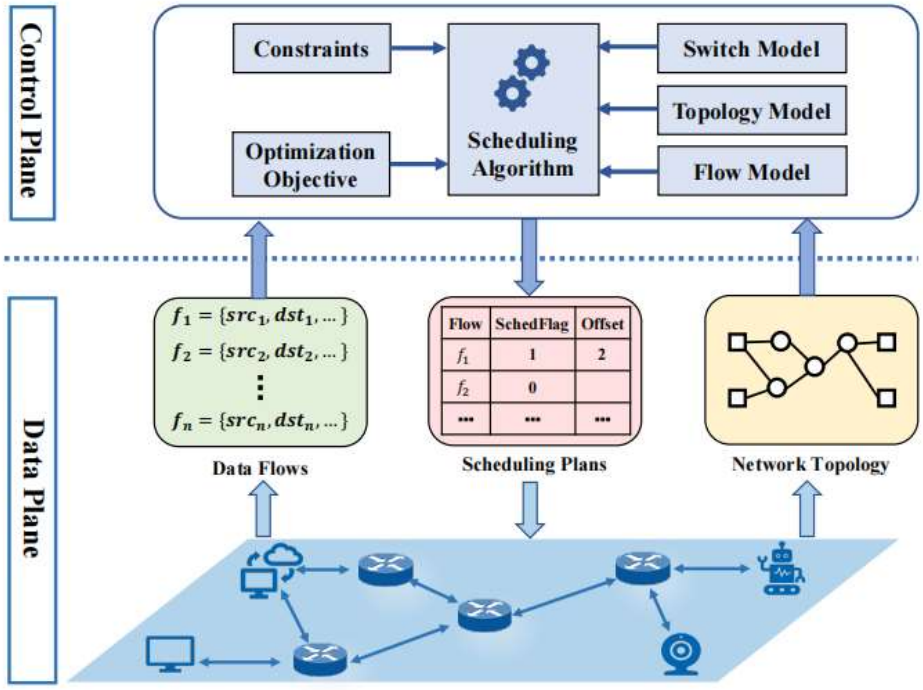

Fig. 1 Scheduling architecture based on CQF

2. Delay-sensitive Traffic Scheduling Model based on CQF

2.1 Switching Model Supporting CQF

The frame forwarding scheme for CQF is shown in Fig. 2 [3]. Time is divided into equally long time slots, and the allocation of time slots follows the following two principles:

- If a switch receives a frame in a time slot, it should be forwarded to the next hop in the next time slot.
- Frame transmission must be completed within the time slot between two adjacent switches.

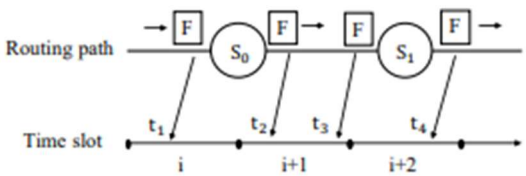

Fig. 2 The frame transmission in CQF

As a result, data transmission in TSN can achieve bounded delay and jitter. In general, the maximum end-to-end delay and the minimum end-to-end delay of frames are represented by:

$$\text{maxDelay} = (\text{hop}_{\text{num}} + 1)\text{T}_{\text{slot}} \tag{1}$$

Data is divided into time slots of equal length, and a frame received in time slot i transmitted in time slot $i + 1$.

$$\text{minDelay} = (\text{hop}_{\text{num}} - 1)\text{T}_{\text{slot}} \tag{2}$$

ISSN: 2414-1895 DOI: 10.6919/ICJE.202401_10(1).0019

Where hop_{num} is the number of switches on the frame's routing path, and T_{slot} is the time slot length.

The switching model supporting CQF is shown in Fig. 3. All elements are synchronized in time. Each port has two queues responsible for receiving and forwarding time-sensitive flows. In one time slot, one queue is in receive mode while the other is in forwarding mode. In the next time slot, the modes of the two queues are reversed. The modes of the two queues are controlled by R_X GCL and T_X GCL. The content of GCLs is fixed, and with time, they flip between 0 and 1 to avoid additional computation and configuration overhead.

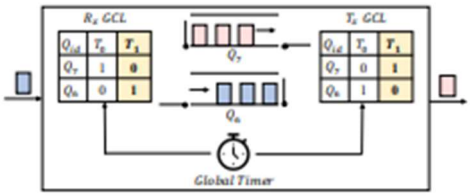

Fig. 3 The switching model in CQF, current time slot is T_1

2.2 Network Model

The network topology in the data plane can be abstracted as a graph $G = (V, E)$, representing a graph composed of vertices and directed edges. The set of directed edges E represents the transmission links in the network, and the set of vertices V consists of hosts and switches, denoted as $V = (H, S)$. Each port of the switch carries 8 queues, with 2 used for buffering time-sensitive flows.

The host is the source and destination of the flow. It generates the data flow and determines when to send the data flow. Each host configures the start time slot for each flow. When a flow cannot be scheduled due to congestion and resource overflow, the host will not send frames belonging to that flow.

In TSN, the minimum interframe spacing for frames belonging to the same flow is the period of the time-sensitive flow. The amount of data sent each period is fixed. A flow sends one frame per period, and data is generated at time 0 (a flow with multiple frames can be considered as multiple single-frame flows). We use F to represent a set of time-sensitive flows. In F, there are n time-sensitive flows. For time-sensitive flow f_i, we can use a 7-tuple to describe it, including the data source, destination, period, data frame size per period, routing path, maximum allowable end-to-end delay, and the offset of the flow relative to the source.

$$\forall \mathrm{f_i} \in \mathrm{F}, \mathrm{i} \in [0, \mathrm{n}-1], \mathrm{f_i} = \{\mathrm{src}, \mathrm{dst}, \mathrm{period}, \mathrm{size}, \mathrm{path}, \mathrm{deadline}, \mathrm{offset}\} \tag{3}$$

Using $f_{i,j}$ to represent the $j+1$th frame in flow f_i (with j starting from 0). The offset is the time from which the first frame of the flow is sent from the source host [4]. The transmission time of frame $f_{i,j}$ at the source node is $\boldsymbol{f_i}$. .offset+j*$\boldsymbol{f_i}$.period. In the problem statement, the offset is a decision variable, while all other properties of f_i are considered priori.

2.3 Problem Model

When assigning offsets to time-sensitive flows, their transmission must comply with constraints within the scheduling hyper-period, which is the least common multiple (LCM) of all flow periods. We use T_{sched} to represent the scheduling hyper-period.

$$\mathrm{T_{sched}} = \mathrm{LCM}(\mathrm{F}, \mathrm{persiods}) \tag{4}$$

ISSN: 2414-1895 DOI: 10.6919/ICJE.202401_10(1).0019

The proposed constraints and optimization objectives are as follows:

Offset constraint.

Offset constraint means that start time of a data flow must fall within the time slot [5]. Otherwise, multiple frames of the same flow will accumulate in the host, consuming limited buffer space. Furthermore, the offset constraint limits the search space for the offset to reduce the complexity of the search.

$$\forall f_i \in F, i \in [0, n-1] \tag{5}$$

$$0 \leq f_i, \text{offset} < \frac{f_i.\text{period}}{T_{slot}} \tag{6}$$

Where T_{slot} is the length of the time slot. Since the granularity of the offset is a time slot, the period $f_i.persiod$ of the flow must be divisible by the length of the time slot.

Time slot constraint.

The time slot constraint limits the maximum and minimum length of the time slot. Since the unit of the offset is the time slot in CQF, it is necessary to ensure that the period of all flows is divisible by the length of the time slot. Therefore, the maximum length is the greatest common divisor (GCD) of all flow periods.

$$\text{Max}(T_{slot}) = \text{GCD}(F.\text{persiods}) \tag{7}$$

Based on the CQF forwarding principle, a frame must be sent and received in the same time slot. Therefore, the length of the time slot should be sufficient to complete frame forwarding. In the worst case, the queue is full, and all frames must arrive at the next hop in the same time slot as they are sent.

$$\text{Min}(T_{slot}) = \frac{\text{Queue}_{\text{length}}}{BW} + \text{hop}_{\text{delay}} + \delta \tag{8}$$

Where $Queue_{length}$ is the maximum number of bytes each queue can accommodate. BW is the link bandwidth, hop_{delay} is the internal processing and propagation delay of the switch, and δ is the clock synchronization precision. In practical applications, $\text{Min}(T_{slot})$ is much smaller than $\text{Max}(T_{slot})$. In CQF-based TSN, T_{slot} is a property of the network and needs to be predetermined as a constant.

Deadline constraint.

In the data flows model, each flow has a deadline [6]. If the start time of a flow is delayed, the flow may not arrive at the destination on time. To avoid this, this constraint limits all traffic to reach their destination before the deadline.

$$\forall f_i \in F, i \in [0, n-1]$$
$$f_i.\text{offset} + \text{hop}_{\text{num}}(f_i) < \frac{f_i.\text{deadline}}{T_{slot}} \tag{9}$$

Where the $hop_{num}(f_i)$ is the number of switches on the path of flow i.

Queue resource constraint.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202401_10(1).0019

A resource block is the queue resource of a port on a switch in a time slot. We use $Q_{S(j,k)}^{T(t)}$ to represent the resource block of port k on switch j at time slot t. Each resource block can be occupied by multiple flows. We represent the occupation status of the resource block by flow as $O\left(f_i, Q_{S(j,k)}^{T(t)}\right)$. If flow f_i occupies resource block $Q_{S(j,k)}^{T(t)}$, then the value of $O\left(f_i, Q_{S(j,k)}^{T(t)}\right)$ is 1, otherwise, it is 0.

$$\forall f_i \in F, i \in [0, n-1], j \in [0, m-1]$$
$$\forall k \in [0, S_j.P_{num} - 1], \forall t \in \left[0, \frac{T_{sche}}{T_{slot}} - 1\right]$$
$$O\left(f_i, Q_{S(j,k)}^{T(t)}\right) = 1$$
s.t.
$$S(j,k) \in f_i.path, \alpha \in [0, \frac{T_{sched}}{f_i.persiod} - 1]$$
$$t = (f_i.\text{offset} + \frac{\alpha * f_i.\text{persiod}}{T_{slot}} + \text{hop}(S_j, f_i) \text{mod} \left(\frac{T_{sched}}{T_{slot}}\right) \quad (10)$$

Where m is the number of switches in the network, $S_j.P_{num}$ is the number of ports on switch j, $S(j,k)$ is the k port of switch j. $hop(S_j, f_i)$ starts from 0, representing which hop switch j is on the path of flow i. α represents which cycle the frame belongs to. t is the time slot occupied by the flow in T_{sche} .

Table 1. Characteristics of flows

Flow	Size	Period (slot)	Routing path
f_1	25	2	$port_A \longrightarrow port_B$
f_2	26	4	$port_A \longrightarrow port_C$
f_3	27	3	$port_A \longrightarrow port_D$

3. Design of flow Scheduling Algorithm based on Mapping Score

To schedule flows as much as possible and achieve load balancing, we need to combine flow sorting and offset searching, considering both the characteristics of flows and the availability of resources, to avoid flow congestion. In this section, we first introduce the Mapping Score to quantify the impact of each flow's scheduling on network resources and explain its necessity. Based on this, we conduct a utility analysis using the Mapping Score for the method that combines flow sorting and offset searching. Finally, we provide a detailed scheduling algorithm based on the Mapping Score.

3.1 Mapping Score

To demonstrate the necessity of considering flow characteristics and resource availability comprehensively, we take scheduling of three flows as an example. Table 1 summarizes the characteristics of the three flows. The queue length of switches is set to be 60.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202401_10(1).0019

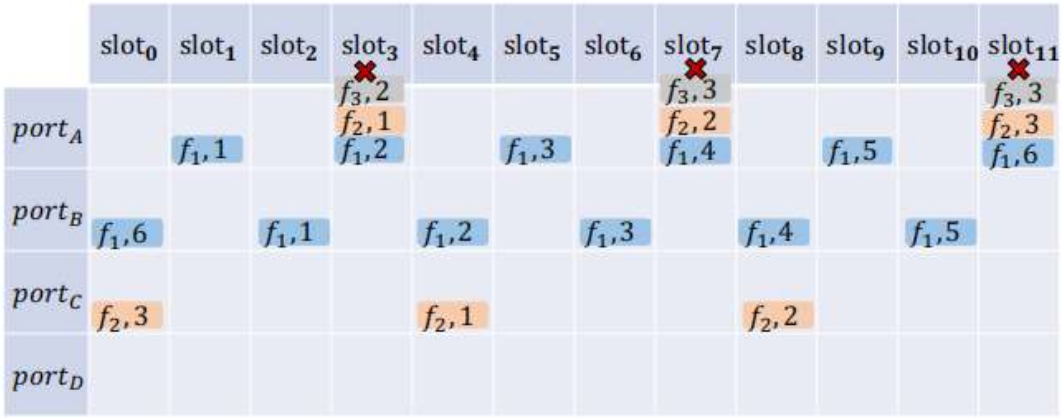

Fig. 4 Scheduling Gantt Chart with traditional greedy algorithm

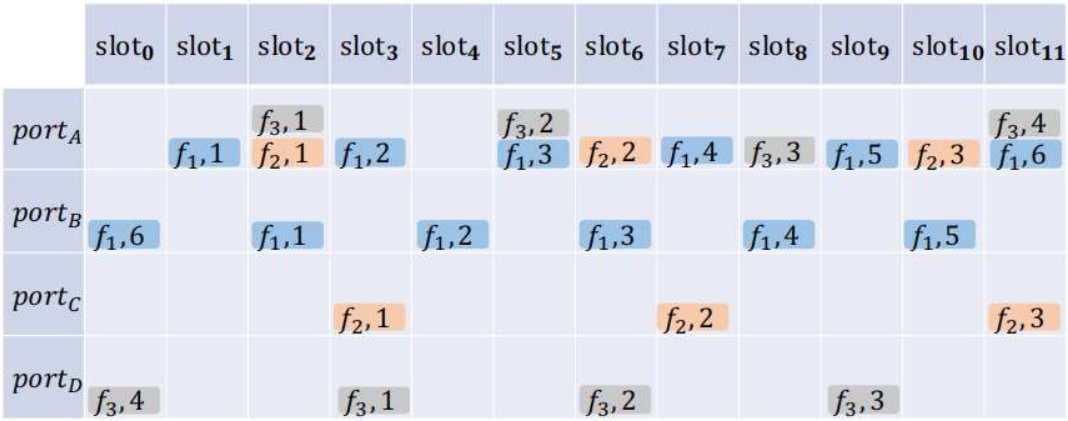

Fig. 5 Scheduling Gantt Chart with Mapping Score

The numbers inside each rectangle in the figure represent the frame number, such as f_1, 1 representing the first frame of flow f1. As shown in Figure 4, the traditional greedy algorithm first sorts the flows based on the frame size. Flows that occupy fewer resources will be mapped first. As a result, the flow sequence is ①f_1, ②f_2, ③f_3. When conducting offset searching, the algorithm maps the selected flows from the largest offset and reserves relatively earlier time slots for the flows mapped later.

When scheduling flows, we consider the selection of a flow and its corresponding offset (i.e., the time it starts from its source) as a scheduling step. For example, the assignment of an offset of 1 for flow f_2 can be represented as $(f_2, 1)$. According to the traditional incremental scheduling greedy algorithm, the scheduling steps are ①$(f_1, 1)$, ②$(f_2, 3)$. Scheduling flow f_3 would result in resource overflow at $slot_3$, $slot_7$ and $slot_{11}$ on $port_A$. Therefore, this algorithm can only schedule two flows.

However, when flow sorting and offset searching are combined, and the characteristics of flows and resource availability are considered simultaneously, the scheduling performance can be improved. We use mapping score to quantify the impact of flow scheduling on network resources. When scheduling flows, we tend to choose scheduling combinations $(flow, \text{offset})$ with the highest mapping scores to minimize the impact on network resources. We first define the mapping score and then provide an example of the scheduling process based on mapping scores in each scenario.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202401_10(1).0019

The resource block at the k-th port of the j-th switch at time slot t is denoted by $Q_{S(j,k)}^{T(t)}$. If a flow occupies a resource block, then $O\left(f_i, Q_{S(j,k)}^{T(t)}\right)$ is set to 1; otherwise, it is set to 0. We define the flow set $F_{S(j,k)}^{T(t)} = \{f | O\left(f_i, Q_{S(j,k)}^{T(t)}\right) = 1\}$, which means that the flow set $F_{S(j,k)}^{T(t)}$ is composed of flows that occupy the same resource block $Q_{S(j,k)}^{T(t)}$. The available resources of $F_{S(j,k)}^{T(t)}$ are denoted as $Res_{S(j,k)}^{T(t)}$.

$$\mathrm{Res}_{S(j,k)}^{T(t)} = \mathrm{Queue}_{\mathrm{length}} - \sum_{f \in F_{S(j,k)}^{T(t)}} f.\mathrm{size} \tag{11}$$

Table 2. Mapping score generated in the second scheduling step

Combination	*Mapping Score*
$(f_2, 3)$	$\frac{60-25}{26}$
$(f_2, 2)$	$\frac{60}{26}$
$(f_2, 1)$	$\frac{60-25}{26}$
$(f_2, 0)$	$\frac{60}{26}$
$(f_3, 2)$	$\frac{60-25}{27}$
$(f_3, 1)$	$\frac{60-25}{27}$
$(f_3, 0)$	$\frac{60-25}{27}$

For each combination $(f_i, offset)$ of a flow and its possible offset, there is a mapping score, calculated as follows:

$$\text{Mapping Score} = \frac{\min \mathrm{Res}_{S(j,k)}^{T(t)} + f_i.\mathrm{size}}{f_i.\mathrm{size}}, \ O\left(f_i, Q_{S(j,k)}^{T(t)}\right) = 1 \tag{12}$$

Where $Min\, Res_{S(j,k)}^{T(t)} + f_i.size$ represents the minimum available resources on the routing path before scheduling f_i. It can only be computed after the offset of f_i is determined, as we need the offset to locate the resource blocks occupied by f_i. A larger mapping score implies a smaller impact of the flow on the network. Therefore, we prioritize the scheduling of the combination $(flow, \mathrm{offset})$ with the highest mapping score.

Furthermore, to detail the scheduling process based on mapping scores, we calculate the mapping scores for all possible scheduling combinations in the given scenario. Since the available resources for all resource blocks are 60, smaller flow sizes result in higher mapping scores. Therefore, we choose to schedule flow f_1 first. Both 0 and 1 are possible values for f_1's offset. We choose 1 as its offset, leaving earlier time slots for other flows. Then, we list all the mapping scores for possible combinations in Table 2. We select the combination $(f_2, 2)$ as the current scheduling step, as it has the highest mapping score. Similarly, we schedule flow f_3 and set its offset to 2. As shown in Figure

ISSN: 2414-1895 DOI: 10.6919/ICJE.202401_10(1).0019

4 and 5, using the mapping score method, we are able to successfully schedule all three flows. Guided by mapping scores, the exemplary scheduling process is①$(f_1, 1)$, ②$(f_2, 2)$, ③$(f_3, 2)$.

3.2 Algorithm Design based on Mapping Score

Mapping time-sensitive flows to lower-level queue resources is equivalent to the bin-packing problem and is an NP-hard problem. In order to improve scheduling performance while reducing complexity, a heuristic algorithm based on mapping scores is proposed, as shown in Algorithm 1. In each iteration, the selection of flows and the determination of their offsets are performed simultaneously. We represent this combination as $(flow, \text{offset})$. In each iteration, the combination with the highest mapping score is chosen as the current scheduling step.

Algorithm 1 *Mapping Score* Based Scheduling (MSS) Algorithm

Input: Flow set F, Network topology G
Output: $F.offset$

1: Initialize available queue resource $Q[port, t]$
2: Initialize mapping score table $Score[f, offset]$
3: **for** $i = 1 : F.num$ **do**
4: **for** each $f_i \in F$ **do**
5: **for** each $offset \in [0, f_i.period - 1]$ **do**
6: $latency = count\_e2e\_latency(f_i, offset)$
7: **if** $latency > f_i.deadline$ **then**
8: $Score(f_i, offset) = -1$
9: // Deadline constraint cannot be met
10: **else if** $QueueOverflow(f_i, offset, Q) == true$ **then**
11: $Score(f_i, offset) = -1$
12: // Resource constraint can not be met
13: **else**
14: $Score = computeMappingScore(f_i, offset, Q)$
15: **end if**
16: **end for**
17: **end for**
18: $[f, offset] = findMaxScore(Score)$
19: $f.offset = offset$
20: $f.schedFlag = SUCCESS$
21: $updateQueueResource(Q, f, offset)$
22: $Score(f, :) = -1$
23: Update score table $Score[f, offset]$
24: **end for**

In lines 4-17, the mapping score for each $(flow, \text{offset})$ combination is updated. Line 5 satisfies the Offset Constraint, lines 6-7 satisfy the Deadline Constraint. Line 10 checks if the Queue Resource Constraint is met. Line 14 calculates the mapping score for each feasible $(flow, \text{offset})$ combination. If the mapping score for the $(flow, \text{offset})$ combination is -1, then the offset value in the combination is not suitable for scheduling the flow f. After updating the mapping score, the combination with the highest score is chosen as the current scheduling step, scheduling the flow f and setting its start time on the host to the offset value. Then, due to flow scheduling, the resource pool is updated in line 20. The Score table is updated in line 23, as all combinations of the scheduled

ISSN: 2414-1895 DOI: 10.6919/ICJE.202401_10(1).0019

flow f are no longer considered in the next iteration. If there are no available flows to schedule, the algorithm ends. The MSS heuristic algorithm generates approximate optimal solutions by fully utilizing knowledge of flow and resource availability. The time complexity of the MSS algorithm is $O(n^2)$. Experimental results show that the scheduling success rate using the MSS algorithm is on average 31.69 % higher than the Naive algorithm, and 4.57% higher than the state-of-the-art FLJ algorithm. Particularly in large-scale linear topologies, its performance is 7.62% higher than existing algorithms. At the same time, it saves more than half the time compared to the computationally expensive taboo-based heuristic algorithm. The MSS algorithm exhibits high time efficiency and scalability in complex network scenarios.

4. Conclusion

This paper addresses the inflexibility issue in traffic scheduling caused by the simple use of flow sorting, offset search, and resource judgment in the traditional cyclic queue and forwarding (CQF). It proposes a delay-sensitive traffic scheduling mechanism based on cyclic queuing and forwarding, including: in the inflexible incremental scheduling, a model of interdependent flow sorting and offset search steps is constructed to enhance the scheduling capability of delay-sensitive services; a unified metric model is designed to quantify the schedulability of each flow sorting and offset search combination strategy, determining the scheduling priority. Mapping score can help the scheduling process maintain load balance at each step and improve the overall load balancing level; a scheduling algorithm based on mapping scores is designed as a general solver, which can generate better flow sorting and offset search combination sequences and ensure global resource load balancing.

References

[1] LIU Yang;LI Zeya;Gong Longqing;XU Danni;TANG Jinfeng. The state-of-the-art research for time-sensitive network and future research interests [J]. Microelectronics & Computer,2022,39(6):1-11.

[2] OpenTSN: an open-source project for time-sensitive networking system development. Wei Quan;Wenwen Fu;Jinli Yan;Zhigang Sun.CCF Transactions on Networking,2020.

[3] YIN Shuwen;WANG Shuo;HUANG Tao. Analysis and Optimization of Queues Based on Network Calculusin Time-Sensitive Networking, 2022,28(1):21-28.

[4] Zhang Xiangwu;Han Dantao;Gong Yanjie. Forecast of the Development of OT Network Based on TSN[J]. China Instrumentation,2023(3):23-26.

[5] DENG Qi-fu. Ultra Reliability--The Introduce of 802.1 Qci in Time-sensitive Networking [J]. Auto Electric Parts2023(1):40-42.

[6] ZHANG Peng;GONG Longqing;XU Danni. A low-overhead wireless TSN clock synchronization method based on beacon [J]. Microelectronics & Computer,2022,39(10):80-87.