Intra-Domain Heuristic Traffic Scheduling Algorithm for Time-Sensitive Networks

Zang YuHang\textsuperscript{1,a}, Wang ZhiLi\textsuperscript{1,b}

\textsuperscript{1}State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China
\textsuperscript{a}yzhang0907@foxmail.com, \textsuperscript{b}jlwang@bupt.edu.cn

Abstract. Time-Sensitive Network (TSN) is a new network transmission technology, which has the advantages of high data transmission and priority setting. Researching and improving the performance of TSN is of great significance to the development of future industrial communication and automation control fields. However, in practical applications, due to complex scenarios and difficulty in TSN traffic scheduling, traditional network traffic scheduling algorithms are not fully applicable to TSN, which may lead to timeout and inefficiency of traffic scheduling, and it is difficult to guarantee the quality of scheduling. In response to this situation, a routing and scheduling algorithm suitable for TSN is proposed. Firstly, TSN is modeled and analyzed, and the balanced routing algorithm based on bi-directional A* algorithm (BR-BidA*) and the modified greedy randomized adaptive search procedures (MGRASP) are proposed to solve the TSN routing and scheduling respectively. Compared with the contrast algorithm, the simulation results show that the proposed method can improve the performance of the algorithm, and greatly improve the solution quality and scheduling success rate.

1. Introduction

With the rapid development of information technology, communication subjects are no longer limited to people or people and things, and communication services that take things and things as entities are increasing day by day. Faced with the evolution of communication subjects, the sensitivity of each business to time has become more stringent. The concept of Real Time Ethernet (RTE) has emerged as the times require. The application scenarios within this concept all regard deterministic delay as a basic requirement, such as factory automation control, automatic driving and power automation, etc. In the above fields, the delay requirements for data transmission are far beyond the controllable range of traditional Ethernet. The IEEE802 working group put forward the concept of Time-Sensitive Network (TSN) to build a unified data link layer protocol, which can run in different fields with the same structure through standardization, so as to provide real-time data transmission guarantee [1].

Time-Sensitive Network has been developed rapidly because of its high data transmission and priority setting function. It is of great significance to study and improve the performance of TSN for the development of industrial communication and automation control field in the future. For Time-Sensitive Network, the timely and reliable transmission of data is closely related to scheduling algorithms. Although TSN has many advantages, it is difficult to schedule traffic in practical application because of complex scenes. The current research results show that the traffic scheduling problem in TSN is an NP-hard problem. Traditional network traffic scheduling algorithms are not fully suitable for TSN, which may lead to timeout and low efficiency of traffic scheduling, and it is difficult to guarantee the quality of scheduling.
At present, the research on TSN mainly focuses on the generation of time triggered flow message scheduling table and algorithm optimization, and most of the research is still in the stage of theoretical research and simulation. In reference [2], YICES SMT solver is used to generate a reasonable scheduling table, but the time complexity of these algorithms is relatively high. When there are many messages to be scheduled in the network, the time to generate scheduling table will be very long. In reference [3], Laursen's goal of Laursen et al.'s work is to determine the routing of Audio-Video-Bridging (AVB) flows so that all frames are schedulable and to minimize the end-to-end delay. They proposed an adaptive heuristic algorithm for routing optimization of AVB flows in Time-Sensitive Network. In reference [4], Craciunas et al. discussed the computational problem of deterministic scheduling of time sensitive traffic conforming to 802.1Qbv standards. In order to ensure low jitter and deterministic end-to-end delay, the constraints needed to calculate offline scheduling are derived, and the required calculation time is weighted. In reference [5], Durr et al. guaranteed the minimum network delay for time sensitive traffic, and mapped the scheduling problem to the "no wait job shop scheduling problem", and then used the tabu search algorithm to solve it. On the whole, the above studies tried to schedule traffic of TSN in different application scenarios, but there are still some problems, such as low algorithm efficiency, separation of routing and scheduling, and difficulty in balancing the scheduling success rate and network delay. Optimization needs further improvement.

In order to solve the above problems, this paper comprehensively considers the routing strategy and traffic scheduling strategy of time triggered flows, models and analyzes the TSN, so as to optimize the end-to-end transmission delay while ensuring the successful scheduling of time triggered flows. In the process of TSN route selection, considering the constraint between the queue load and the scheduling success rate, the BR-BidA* algorithm is used for planning. In the process of TSN traffic scheduling, this paper proposes a modified greedy randomized adaptive search procedures (GRASP) based traffic scheduling algorithm. In the construction stage, the GRASP algorithm is improved to generate the initial solution; in the local search stage, the initial solution is optimized by local disturbance, and the disturbance strategy is improved. Under the premise of ensuring that the scheduling is reachable, the end-to-end delay of each time trigger flow is optimized to obtain the optimal scheduling scheme.

2. System Models

2.1. Network Model
The abstract representation of TSN model includes hosts, network switches and physical links. The given network topology is represented as a directed graph G, as shown in Eq. (1):

$$G=\{V, E\}$$

Where $V$ is the device node in the TSN, including hosts $H$ and network switches $S$, as shown in Eq. (2):

$$V=H \cup S$$

$E$ represents the set of physical links in the network, and a full-duplex network link in TSN is represented as a tuple, as shown in Eq. (3), where two nodes in tuple $(i, j)$ are connected through a network link.

$$E=\{(i, j)|i, j \in V\}$$

2.2. Traffic Model
In a TSN, the real-time data sequence is modeled as a set of message tuples sent from the source node to the destination node, that is, the flow in the TSN system, represented by set $F$. For each flow $f$, it can be represented by a tuple associated with attributes, as shown in Eq. (4):

$$f=(v_s, v_d, T, D, P)$$

where $v_s$ is the source node of the system sender, $v_d$ is the destination node at the system receiver, $T$ is the transmission period of the flow, $D$ is the transmission time limit from the source to the destination, and $P$ is the data size of the flow. The data size of flow is in bytes, and a single frame can transmit a payload of up to 1500 bytes, which is the so-called maximum transmission unit of Ethernet. If the size of
flow data is larger than the maximum transmission unit, the message will be divided into multiple frames for transmission.

Route $R_i$ is the transmission path of flow $f_i$ in the TSN. $f_i.v_s$ is the source node of $f_i$, $f_i.v_d$ is the destination node of $f_i$. For flow $f_i$ whose source node and destination node are known and passed through multiple intermediate nodes, its route can be expressed as Eq. (5):

$$R_i=\{f_i.v_s,...,f_i.v_d\}$$

TSN provides ultra-low delay and jitter guarantees for time sensitive data. In the network, the mechanism of gate opening and closing is used to control the transmission of data. The GCL (gate control list) is used to control the opening and closing state of each queue at a certain time. Generally speaking, GCL is cyclical. Before and during the transmission of critical data, GCL will stop sending other data to ensure that the critical data is not affected. This paper expects to optimize the end-to-end transmission delay while ensuring the successful scheduling of TSN data flows. In this process, the following constraints should be met. Among them, $Frame_{i,j,k}$ represents the $k$-th frame of flow $f_i$ on physical link $j$. $TS_{i,j}$ represents the allocated time slot of flow $f_i$ when transmission on physical link $j$, $t_{open}$ and $t_{close}$ represent the opening time and closing time of GCL respectively.

Eq. (6) indicates that when mapping data flows to time slots, each flow should be allocated exactly one slot, where $i_1, i_2 \in f_i$, and $i_1 \neq i_2$.

$$TS_{i_1,j} \neq TS_{i_2,j}$$

Eq. (7) indicates that for any network link, frames of different flows are transmitted independently without overlapping each other, where $i_1, i_2 \in f_i$, and $i_1 < i_2$.

$$Frame_{i_1,j,k}.t_{open} \geq Frame_{i_2,j,k}.t_{close}$$

Eq. (8) indicates that for any GCL, the gate closing time must be after the corresponding gate opening time.

$$Frame_{i,j,k}.t_{close} \geq Frame_{i,j,k}.t_{open}$$

Eq. (9) indicates that when data frames are transmitted on the same link, different frames of the same data flow must be scheduled in their order according to the GCL.

$$Frame_{i,j,k+1}.t_{open} \geq Frame_{i,j,k}.t_{close}$$

Eq. (10) indicates that during the complete scheduling period of a data flow, the links that the flow passes through must be in the order according to the routing table $R$, where $j_1, j_2 \in f_i$, and $j_1 < j_2$.

$$Frame_{i,j_2,k}.t_{open} \geq Frame_{i,j_1,k}.t_{close}$$

Eq. (11) indicates that the same data flow must satisfy the relative time limit from the source to the destination in its complete scheduling period. $Frame_{i,j,k+1,first}$ is the first frame of $f_i$ on the first physical link, and $Frame_{i,j,k,last}$ is the last frame of $f_i$ on the last physical link.

$$Frame_{i,j,k,last}.t_{close} - Frame_{i,j,k,first}.t_{open} \leq D_i$$

3. Algorithm Description

3.1. Balanced Routing Algorithm Based on Bi-directional A*

In IEEE 802.1Qca, TSN uses the shortest path algorithm by default. In reference [6], the routing strategy adopts the K-Shortest Paths algorithm(KSP). The algorithm starts from the shortest path and generates K unique routes with increasing lengths. If there are multiple routes with the shortest paths, one of them will be randomly selected according to the heuristic method. Although the routing solution generated by the KSP algorithm optimizes the routing selection to a certain extent, the reference [6] shows that if KSP algorithm is used in TSN, the routing scheme selection based only on the path length will lead to the scheduling timeout problem.

In a TSN, routing according to the shortest path will intuitively reduce the end-to-end delay of a flow. However, in this case, the link load will increase, and there will be mutual influence between flows and unsatisfactory of flow time limits, which will affect the schedulability of flows in the TSN system.

3
Therefore, this paper proposes a balanced routing algorithm based on Bi-directional A* Algorithm [7] (BR-BidA*), which prioritizes link load factors, optimizes the overall delay in routing selection, and improves the success rate of scheduling.

During the execution of the BR-BidA* algorithm, the nodes and links in the TSN are first initialized, and the links are arranged in ascending order according to the load. Then, the first k shortest paths of the current flow are obtained through the BidA* algorithm, and the path set is put into the candidate list. According to the load of the paths in the candidate list, the load optimal path in the list can be obtained and put into the result list. At this time, judge whether the next flow is empty. If it is, the algorithm ends and outputs the routing result list. Otherwise, the next flow will be calculated sequentially until the end of the algorithm. The execution process of BR-BidA* (G, F) algorithm is as follows:

```plaintext
Produce BR-BidA*(G, F)
1: R←\emptyset
2: for \( f_i \in F \) do
3: \( R^{\text{BidA*}} \leftarrow \text{BidA*} (G, f_i.v_s, f_i.v_d, K) \)
4: for \( r \) in \( R^{\text{BidA*}} \) do
5: for \( m \) in range(0, \( K \)) do
6: for \( n \) in range(\( r.size - 1 \)) do
7: load = load + current_load
8: end for
9: load_set.add(load)
10: load = 0
11: end for
12: Route= Min(load_set)
13: Routes.add(Route)
14: r.update(load)
15: end for
16: R←R ∪ Routes
17: end for
18: return R
```

Compared with the Bi-directional A* Algorithm [7], BR-BidA* algorithm calculates the load value of each alternative path of a single flow based on the idea of load balancing, and takes the path with the minimum total load value as the optimal load path to reduce the delay caused by link congestion.

3.2. Modified Greedy Randomized Adaptive Search Procedures Traffic Scheduling Algorithm

The GRASP algorithm [6] is a very suitable heuristic algorithm for solving combinatorial optimization problems. In the construction stage, the initial feasible solution is generated by the greedy random algorithm combining the characteristics of both greedy and random. In the local search stage, the local search algorithm is used to improve the initial solution and find the optimal solution. However, the random selection of threshold parameters in the construction stage of the GRASP algorithm will reduce the quality of the solution, and the optimization effect will be poor. Moreover, in the local search stage, the GRASP algorithm lacks effective heuristic strategies when solving the TSN traffic scheduling problem, and it is easy to fall into a local optimum prematurely, which affects the quality of the solution.

Aiming at the traffic scheduling problem in TSN, a modified greedy randomized adaptive search procedures (MGRASP) scheduling algorithm is proposed in this paper, as shown in Figure 1.
Initialization threshold parameter list and probability list, maximum iterations and search rules

Begin

Selecting threshold parameters based on probability list

Constructing initial solution based on threshold parameters

Carried out local disturbance in the local search phase

Accept neighborhood solution

Update threshold parameter list value and threshold usage frequency

Is the current iteration cycle over?

Is the maximum number of iterations reached?

End

Figure.1 MGRASP Algorithm

3.2.1. Construct Initial Solution

In the MGRASP algorithm, a greedy function is defined, and the threshold parameters are dynamically updated by the evaluation value of the greedy function in each iteration, and then the restricted candidate list (RCL) is constructed by selecting elements. On the basis of ensuring the quality and randomness of the initial solution, an element is randomly selected from the RCL to join the current local solution, and the element is deleted from the scheduling scheme set. After each element is selected from the RCL, the construction cost of the remaining elements is evaluated by the greedy function, and the RCL is updated. Repeat the process until all elements are processed, and then terminate the iteration. The algorithm in the construction stage is as follows:

Produce MGRASP_Construction(G, F, R, A)

1: x← ∅
2: F'←SortByPeriod(F,R)
3: for f_i ∈ F' do
4: α_k=Select(A,P)
5: RCL←ConstructionSolution (α_k, f_i)
6: while ScheduleFlow(x, f_i, RCL)=true do
7: x←x ∪ {f_i}
8: RLC. Add (x')
9: end while
10: if RCL≠ ∅
11: x←RCL.GetRandom()
12: end if
13: end for
14: return x

3.2.2. ScheduleFlow Algorithm

The ScheduleFlow algorithm is used to determine whether a single flow is successfully scheduled within the time limit. The algorithm steps are as follows:

Step 1: on the current idle link, arrange the first frame on the route according to the earliest offset;
Step 2: determine whether the frame can be allocated to an available time slot other than the previous link. If so, the frame will be allocated and step 4 will be executed; otherwise, step 3 will be executed;

Step 3: backtrack the previous frame and randomly shift it backward by $i$ units and rearrange it to an available time slot;

Step 4: determine whether the last frame has been scheduled. If so, step 5 will be performed; otherwise, step 2 will be performed for the next frame;

Step 5: determine whether the last frame is scheduled before the deadline. If so, the scheduling is successful; otherwise, the scheduling fails and the algorithm ends.

### 3.2.3. Adjust the Threshold Parameters Dynamically

The MGRASP algorithm defines a threshold parameters list $A$ and its selection probability list $P$ in the initialization stage, as shown in Eq. (12) and Eq. (13).

$$A = \{a_1, a_2, a_3, \ldots, a_i\} \quad (12)$$

$$P = \{p_1, p_2, p_3, \ldots, p_i\} \quad (13)$$

In each iteration cycle, after the neighbourhood solution is received, the usage frequency of each threshold parameter in the threshold parameter list $A$ is counted, and the corresponding delay optimal solution $Z$ is recorded. After the iteration cycle ends, the average value of $a_i$ in the whole cycle is calculated, as shown in Eq. (14).

$$\text{Avg}_{a_i} = \text{Average}(a_i) \quad (14)$$

The threshold parameter evaluation function is defined in Eq. (15).

$$G(a_i) = \frac{Z}{\text{Avg}_{a_i}} \quad (15)$$

According to the statistical information, the selection probability of each threshold parameter is updated and stored in the selection probability list $P$, as shown in Eq. (16).

$$p_i = \frac{G(a_i)}{\sum_{i=1}^{n} G(a_i)} \quad (16)$$

In the dynamically adjusted threshold list, the selection probability corresponding to the threshold parameter can be used to measure the convergence degree between the solution in the iteration cycle and the current optimal solution. In the process of multiple iterations, according to the adjusted selection probability list $P$, the threshold with the larger probability value will be selected first. In this way, the direction of the algorithm can be continuously optimized and adjusted, which is conducive to improving the quality of the solution.

### 3.2.4. Local Search

In the local search phase, the MGRASP algorithm is improved on the basis of reference [8], and a disturbance strategy is designed to disturb the current local optimal solution, which effectively avoids the GRASP algorithm from falling into the local optimum easily. Aiming at the delay optimization goal of intra-domain scheduling in TSN, a simulated annealing probability acceptance strategy[9] is designed to minimize the scheduling delay, and the new solution after the local search is selected with a specific acceptance probability $p$. This method can guide the local search direction and control the searching process reasonably, which is beneficial to the algorithm to move towards the direction of reducing the scheduling delay and greatly improves the quality of the solution. The algorithm process of local search is as follows:

```plaintext
Produce Local_Search(Solution)
1: while Eval(S.F) >= Eval(Solution.F) do
2: if(e^(-(Eval(S.F)-Eval(Solution.F))/T) > p) then
3: Solution ← s
4: Solution ← Disturbance_Strategy(Solution)
5: end if
6: end while
7: return Solution
```
WCD is the worst-case end-to-end delays of time triggered flows, as shown in Eq. (17).

$$WCD(f_i) = \text{Max}(f_i.t_{close}) - \text{Min}(f_i.t_{open})$$ (17)

The evaluation function Eval evaluates the end-to-end delay of the time triggered flow, as shown in Eq. (18).

$$\text{Eval}(F) = \min(\sum_{f_i \in F}(WCD(f_i)))$$ (18)

4. Experiment and Analysis

In order to verify the effectiveness of this algorithm, an intra-domain TSN and the corresponding data set are constructed for simulation experiments. At the same time, the KSP routing algorithm, the traditional A* routing algorithm, and the proposed BR-BidA* algorithm are compared. Finally, the MGRASP traffic scheduling algorithm is analyzed and compared with the GRASP algorithm in reference [6].

In the hardware environment Intel Core i7-6500U, 8GB RAM, the MyEclipse2014 version of IDE is used for the experiment, and the node matrix of 50×50, 100×100 and 150×150 are used to testing. The test results are shown in Table 1.

| Node Size | KSP      | A*       | BR-BidA* |
|-----------|----------|----------|----------|
| 50×50     | 169ms    | 147ms    | 138ms    |
| 100×100   | 1854ms   | 1423ms   | 662ms    |
| 150×150   | 8928ms   | 6629ms   | 2036ms   |

The experimental results show that in terms of the running time of routing, the BR-BidA* algorithm is the best, the traditional A* routing algorithm is the second, and the KSP routing algorithm is the longest. When performing the intro-domain TSN scheduling, the KSP algorithm is usually used for routing selection. Although this algorithm optimizes the routing selection to a certain extent, but in the scene with a large number of nodes, there are shortcomings such as slow calculation speed, lack of searching directionality, and more useless nodes generated in the intermediate process. Compared with the KSP algorithm, the traditional A* routing algorithm has directionality in the routing process, and the efficiency of the algorithm is improved. Based on the traditional A* routing algorithm, the BR-BidA* algorithm searches paths from the source side and the destination side in parallel, which can greatly shorten the running time and improve the performance of the routing algorithm in the scenarios with many nodes.

The experimental results show that the combination of the BR-BidA* algorithm and the MGRASP traffic scheduling algorithm has obvious advantages in the final objective function value. The average value of convergence algebra is 38, and the average objective function value is 15.5ms, as shown in Figure 2.

![Figure 2: Scheduling delay of GRASP and MGRASP algorithm](image-url)
In terms of scheduling success rate, when the number of time triggered flows increases, the combination of the BR-BidA* algorithm and the MGRASP traffic scheduling algorithm can reduce the scheduling failures caused by violations of time slot conflict constraints and end-to-end delay constraints, and improves the scheduling success rate, as shown in Figure 3.

![Figure 3 Scheduling success rate of GRASP and MGRASP algorithm](image)

Compared with the GRASP algorithm, the MGRASP traffic scheduling algorithm slightly increases the running time, but the difference is not significant, as shown in Figure 4.

![Figure 4 Running time of GRASP and MGRASP algorithm](image)

By dynamically adjusting the threshold parameters, the MGRASP traffic scheduling algorithm can optimize the solution direction and improve the feasibility of scheduling. At the same time, in the local search phase, the MGRASP traffic scheduling algorithm adopts a disturbance strategy, which can reasonably control the search process. Although the running time of the algorithm is slightly increased, it is conducive to reduce the scheduling delay, and improves the scheduling success rate.

5. Conclusion
Aiming at the problems of low scheduling success rate and low algorithm operation efficiency in the intra-domain TSN traffic scheduling, this paper proposes the BR-BidA* algorithm and the MGRASP traffic scheduling algorithm. The experimental simulation results show that the BR-BidA* algorithm helps reduce the delay caused by link congestion and improves the performance of the routing algorithm. The MGRASP traffic scheduling algorithm can dynamically adjust the threshold parameters, and continuously adjust the solution direction, which improves the schedulability of TSN traffic. At the same time, the disturbance strategy is used to control the search process, which is beneficial to the algorithm to move towards the direction of reducing the scheduling delay, and ultimately reduces the overall scheduling delay and improves the scheduling success rate.
References

[1] IEEE Official Website of the 802.1 Time-Sensitive Networking Task Group, 2016, [online] Available: http://www.ieee802.org/1/pages/tsn.html.

[2] Steiner W. An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks[C]. Real-Time Systems Symposium (RTSS), 2010 IEEE 31st. IEEE, 2010: 375-384.

[3] Sune Molgaard Laursen, Pop P, Steiner W. Routing Optimization of AVB Streams in TSN Networks[M]. ACM, 2016.

[4] Silviu S. Craciunas, Ramon Serna Oliver, Martin Chmelik, et al. Scheduling Real-Time Communication in IEEE 802.1Qbv Time Sensitive Networks. 2016, :183-192.

[5] Nayak N G. No-wait Packet Scheduling for IEEE Time-sensitive Networks (TSN)[C]/ International Conference on Real-time Networks & Systems. ACM, 2016.

[6] V. Gavriluț, L. Zhao, M. L. Raagaard and P. Pop, "AVB-Aware Routing and Scheduling of Time-Triggered Traffic for TSN," in IEEE Access, vol. 6, pp. 75229-75243, 2018, doi: 10.1109/ACCESS.2018.2883644.

[7] ZHANG Xiaohui, ZHI Baoping. Implementation of an improved bi-directional A* algorithm in urban geographic information system [J]. Science of Surveying and Mapping, 2020, 45(02): 145-149.

[8] Gavriluț V, Zhao L, Raagaard M L, et al. AVB-aware routing and scheduling of time-triggered traffic for TSN[J]. IEEE Access, 2018, 6: 75229-75243.

[9] Kirkpatrick S, Vecchi M. Optimization by simulated annealing [J]. Science, 1983, 220(4598): 671-680