Time-Aware Traffic Scheduling with Virtual Queues in Time-Sensitive Networking

Junli Xue, Guochu Shou, Yaqiong Liu, Yihong Hu, and Zhigang Guo
Beijing Key Laboratory of Network System Architecture and Convergence, School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing, China
{junlixue, gcshou, liuyaqiong, yhhu, gzgang}@bupt.edu.cn

***Abstract*—Time-Sensitive Networking is a promising technology that supports deterministic services with bounded delay, low packet delay variation, and low packet loss. Time-Sensitive Networking consists of several components, in which traffic scheduling plays an important role to provide the delay guarantee. The traffic scheduling on the level of flow is challenging for dense traffic demands. In this paper, we propose a time-aware traffic scheduling scheme based on network virtualization to schedule a large number of flows. In the scheme, the physical queues of a network node are abstracted as virtual queues, which isolate each flow with a time offset parameter. A scheduling algorithm is designed into two stages of virtual queues mapping and schedule computation. The simulation results in a use case of industrial automation show the effectiveness of the proposed scheme.**

***Index Terms*—Time-Aware Scheduling, Time-Sensitive Networking, virtual queues**

I. Introduction

Time-Aware Scheduling (TAS) is one of the key methods of traffic scheduling in TSN [1]. With the TAS method, the TSN node can transmit the traffic at a precise time, providing the delay guarantee for the traffic of deterministic services, namely Scheduled Traffic (ST) . The TAS consists of the gate control mechanism and the scheduling procedure. The gate control mechanism is the hardware mechanism and controls the traffic transmission by setting the gate state of the physical queues. The scheduling procedure runs the algorithm to compute a schedule, which contains the gate state and time interval of maintaining the state. The gate control mechanism is tightly coupled with the TSN node. The scheduling procedure runs in the node or centralized network configuration (CNC) entity, which is dependent on the configuration mode.

TAS is required with flow level scheduling capabilities to provide a bounded delay for each traffic flow. However, the number of physical queues for gate control mechanism is restricted. Multiple flows are inevitably transmitted in the same physical queue, especially in the case of a large number of flows. This introduces the flow isolation problem. Although the gate control mechanism can operate on the granularity of the flow to isolate multiple flows, the flow isolation only depending on physical queues is challenging and complex. On the one hand, the frequent conversion of the gate state can make the node overload. On the other hand, it also increases the number of schedule entries, which are limited in the node.

978-3-903176-32-4

The TAS algorithms attract the most attention of the researchers, and are developed based on the mature solver and heuristic. Craciunas et al. [2] model the TAS problem as an optimization problem and solve it with Satisfiability Modulo Theories (SMT) and Optimization Modulo Theories (OMT) solver. Dürr and Nayak [3] model the TAS problem as a no-wait packet scheduling problem and propose a tabu search algorithm to compute the schedule of flows. Gavrilut and Pop [4] consider the scheduling of ST flows, Audio and Video Bridging(AVB) flows, and best-effort flows. They propose a greedy randomized adaptive search algorithm, which guarantees the requirements both of ST flows and AVB flows.

In this paper, we propose solving the flow isolation issue with network virtualization, which can allocate resources flexibly without restricted by the granularity of physical resources. We present a TAS scheme with virtual queues (VQ-TAS), in which the virtual queues isolate traffic flows with a time offset parameter. We formulate the scheduling problem and design a scheduling algorithm including virtual queues mapping and schedule computation. The effectiveness of the proposed scheme is evaluated in a use case of the industrial automation applications from the number of schedulable flows, the end-to-end delay, and the runtime of the algorithms.

II. VQ-TAS Scheme

The VQ-TAS scheme is shown in Fig. 1. The output port has multiple physical queues with transmission gates, and the gates are open/closed at a precise time according to a schedule, namely Gate Control List (GCL). Only when the transmission gate of the queue is open and there is no traffic in the high-priority queue, the traffic can be transmitted. The physical queue follows the first-in first-out (FIFO) rule to transmit traffic. The ST flows store in the virtual queue, and then enter the allocated physical queues through the virtual queue mapping. The mapping table of the virtual queue mapping represents the relationship between virtual queues and physical queues, including the sequence of flows in the same physical queue. In the context of FIFO, the order in which the flows enter into the physical queue is the order in which flows are transmitted. This determines the time offset of each flow relative to the flow at the head of the queue. Time offset makes it possible to avoid overlapping between flows within a physical queue.

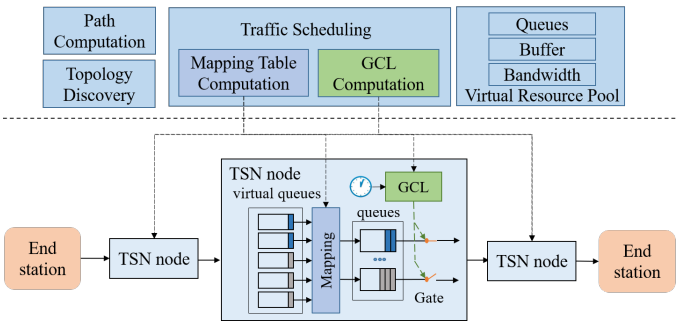

Fig. 1. Illustration of the VQ-TAS scheme.

With the centralized configuration mode, the configuration entity such as CNC is responsible for network control and configuration. As showed in Fig. 1, the core modules include topology discovery, path computation, a virtual resource pool, and traffic scheduling. The topology discovery module gives a global view of the network and the path computation module computes the path with the global view. The virtual resource pool includes the bandwidth, buffer, and queues abstracted from the physical network. The traffic scheduling module consisting of mapping table computation and GCL computation submodules schedule the traffic flows with the virtual resource.

In detail, the scheduling procedure sets up by a report of the TSN node. The report includes the source node, destination node, period, and length of the ST flows. The deadline requirement is from the service request. The Mapping Table Computation submodule generates mapping tables with the flow set, paths and queues resource from those reports, the path computation module and the virtual resource pool, respectively. After that, the GCL computation submodule runs the algorithm and generates GCLs of physical queues. Finally, the mapping tables and GCLs are delivered to each node of the path.

III. Models and Problem Formulation

A. Models

The network is modeled as a graph $G = (V, E)$, where V is the set of network nodes, and E is the set of network links. A link $[v_i, v_j]$ of E connects the source node v_i and the destination node v_j. The set of physical queues in node v_i is denoted as Q^i, $Q^i = \{q_m^i \mid v_i \in V\}$. The virtual queues set is denoted as VQ^i, $VQ^i = \{vq_n^i \mid v_i \in V\}$.

F is defined as the set of ST flows. The flow $f_k \in F$ is characterized by a five-tuple $(src_k, dst_k, dl_k, p_k, l_k)$, where src_k is the source node, dst_k is the destination node, dl_k is the deadline, p_k is the period of the flow, and l_k is the flow length. In this paper, we define the bytes of the flow per period as flow length. The flow should reach the destination node before its deadline. Without loss of generality, all parameters are positive integers. The path of the flow f_k is defined as $path_k = [src_k, v_1], [v_1, v_2], \ldots, [v_{i-1}, v_i], [v_i, dst_k]$.

Traffic flows experience an end-to-end delay transmitting along the path. The end-to-end delay of a flow can be divided into four parts of the propagation delay, serialization delay, forwarding delay, and queuing delay [5]. The end-to-end delay ed_k of flow f_k transmitting along $path_k$ is given as

$$ed_k = \sum_{[v_{i-1},v_i] \in path_k} pd_k^{[v_{i-1},v_i]} + sd_k^{src_k} + sd_k^{dst_k} + \sum_{v_i \in V_k'} \left(2sd_k^i + fd_k^i + qd_k^i\right). \tag{1}$$

where $pd_k^{[v_{i-1},v_i]}$ denotes the propagation delay, sd_k^i denotes the serialization delay, fd_k^i denotes the forwarding delay, qd_k^i queuing delay, and V_k' denotes all switching nodes of the $path_k$. Serialization is the process of transmitting a flow from the network node buffer to the link and deserialization is inverse [5]. The serialization delay is multiplied by 2 because of the serialization and deserialization processes in the switching node.

With the gate control mechanism, the queuing delay of the flow is controlled by the transmission gate. The gate open time of the queue q_m^i is denoted as ot_m^i, the open window is denoted as ow_m^i, and the gating cycle of the GCL is denoted as gc^i. For ST flows, the cycle of each queue is the period of the flow, and the cycle of the GCL is the least common multiple of the cycle of each queue. Denote the time that the flow f_k arrives in the queue q_m^i as $at_{m,k}^i$, the queuing delay is given as

$$qd_k^i = (ot_m^i - at_{m,k}^i) + \sum_{at_{m,r}^i < at_{m,k}^i} sd_r^i. \tag{2}$$

The queuing delay consists of two parts, i.e., the delay waiting for the gate to be open and the delay waiting for the flows arrived before to finish their transmission.

In the VQ-TAS scheme, the queuing delay of each flow in the node is determined by the weight mapping matrix W^i generated by the virtual queue mapping and the GCL. The weight is the order in which the flows enter the physical queue. The $W^i(n, m) = x$ indicates that the vq_n^i is mapped to the q_m^i and the flow in vq_n^i is the xth flow that enters physical queue q_m^i. The order determines the time offset of each flow from the flow at the head of the queue, which is denoted as off_k^i. The queueing delay is given as

$$qd_k^i = (ot_m^i - at_{m,k}^i) + off_k^i, \tag{3}$$

$$off_k^i = \sum_{r=1}^{k-1} sd_r^i * W^i(n, m), f_k \to vq_n^i. \tag{4}$$

B. Problem Formulation

Given the ST flows and their paths, the TAS problem is to determine the schedule for each flow such that the deadline requirement of each flow is satisfied.

1) Objective function: We formulate the TAS problem as an optimization problem with a goal of minimizing the end-to-end delay of all flows, i.e., minimizing the maximum end-to-end delay of the flow set. The smaller the end-to-end delay, the smaller the network resources are occupied by a flow, then

the more ST flows can be scheduled. The objective function is as following:

$$\min\{\max\{ed_k|f_k \in F\}\}$$

2) Scheduling constraints: If the flow cannot finish transmission in an open window, the flow cannot be selected for transmission. The flow waits to transmit in the next open window or even next gating cycle, resulting in the delay variation. The open window constraint is given as,

$$\forall v_i \in V, \forall f_k \in F : \frac{l_k}{B_m^i} \leq ow_m^i \leq gc^i, \tag{5}$$

where B_m^i is the allocated bandwidth for queue q_m^i.

In order to avoid flow interference among multiple physical queues, the open windows of their gates are not allowed to overlap. The flow isolation constraint of multiple queues is given as

$$\begin{aligned} &\forall v_i \in V, \forall f_k \in F, f_j \in F, k \neq j : \\ &f_k \rightarrow vq_n^i \rightarrow q_m^i, f_j \rightarrow vq_{n'}^i \rightarrow q_{m'}^i : \\ &\left(ot_m^i \geq ot_{m'}^i + ow_{m'}^i\right) \cup \left(ot_{m'}^i \geq ot_m^i + ow_m^i\right). \end{aligned} \tag{6}$$

Flow interference also should not occur in the same physical queue. Since the flows of the identical physical queue are transmitted with a time offset in the VQ-TAS scheme, the flows are isolated.

Two flows cannot be transmitted on the same link at the same time. In the VQ-TAS scheme, the flows in the same queue are transmitted sequentially. Therefore, link constraint releases and requires the flows in different queues should not be transmitted at the same time. With the flow isolation constraint, the link constraint is satisfied.

The deadline constraint is given as

$$\forall f_k \in F : mspan_k \leq dl_k. \tag{7}$$

where $mspan_k = gt_k + ed_k$ is the time that flow f_k finishes transmission and gt_k is the time that the source node generate flow f_k.

C. VQ-TAS Algorithm

The VQ-TAS algorithm is to get the weight mapping matrix and the GCLs such that the deadline of each flow is satisfied. The algorithm is divided into two stages of virtual queues mapping and GCL computation. The stage of virtual queues mapping allocates the virtual queues for each flow and maps the virtual queues to physical queues. If the number of physical queues is smaller than that of virtual queues, the virtual queues are clustered. Then each cluster is allocated with a physical queue. The virtual queues in one cluster are sequenced with the merge sort algorithm. This stage generates a mapping matrix for GCL computation. The stage of GCL computation computes the content of GCL for physical queues. The GCL is got with an ILP solver.

Algorithm VQ-TAS algorithm

Input: Flows set F, Path set $Path$, Physical queues set Q
Output: Mapping matrix W, Time schedule S

```
W ← {}, S ← {}
Compute the queue delay bound of each flow in each node
for i = 1 → |V| do
    W^i ← 0_{n×m}, S^i ← {}
    Allocate virtual queues for each flow
    Split virtual queues into cluster according to queue delay bound
    Sort virtual queues in the same cluster with merge sort algorithm
    Allocate physical queues for each cluster
    Generate W^i
    for m = 1 → |Q^i| do
        find the solution S_i = (ot_m^i, ow_m^i) by a ILP solver with the objective function and scheduling constraints
    end for
    W ← W^i, S ← S^i
end for
```

IV. A Case Study

A. Simulation Setup

We evaluate the performance of the VQ-TAS scheme in a use case of industrial automation. A chain topology consisting of IO stations, an industrial controller, and TSN nodes is selected, which is common in industrial automation. The isochronous traffic from the IO station to the controller is selected as ST. The cycle of the ST flows is 100 μs-2 ms, and the ST flow needs to finish transmission in one cycle [6]. The propagation delay is set to 0.5 μs, and the forwarding delay is set to 1 μs. The bandwidth is set to 1 Gbps and 10 Gbps, respectively. The flow length set is {64 bytes, 256 bytes, 1518 bytes}. The probability that the path of each flow is 3/4/5/6/7 hops is 0.1/0.1/0.1/0.3/0.4 [7]. All ST flows have the same period. The scheduling algorithms are implemented in a 64 bit 8-core 3.4 GHz Intel Core-i7 PC with 16GB memory. The ILP tool is PuLP in Python and the SMT solver is Z3 4.8.8.0.

B. Schedulable ST Flows

Fig. 2(a) shows the maximum number of ST flows that can be scheduled with a different deadline. L represents the flow length. The results show that the number of ST flows increases approximately linearly with the deadline and the growth of flow length decreases the number of schedulable flows. The number of schedulable ST flows with 10 Gbps bandwidth is about 10 times than that with 1 Gbps bandwidth, which is the ratio of the two bandwidths. This indicates that the queuing delay is mainly caused by the transmission of other flows in the VQ-TAS scheme. When the flow length is 1518 bytes and the deadline requirement is less than 500 μs, the number of schedulable ST flows is smaller than the 1/10 of that with 1 Gbps bandwidth. This is because the delays of propagation, serialization, and forwarding are too large to

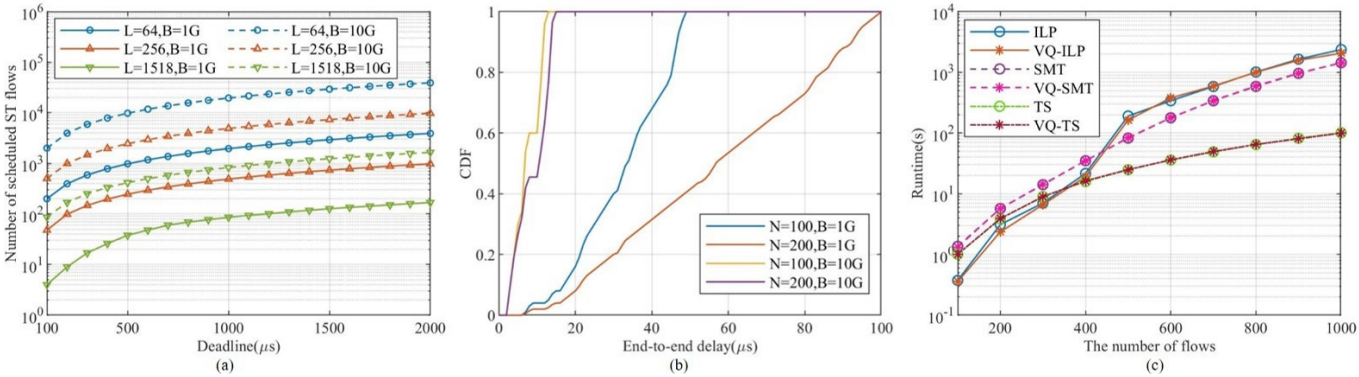

Fig. 2. Simulation results:(a) the number of schedulable ST flows, (b) the cumulative distribution of end-to-end delay, and (c) runtime of the scheduling algorithms w/o virtual queues.

satisfy the deadline. The results demonstrate that the physical resources are the most critical limitation and the VQ-TAS scheme is an optimization of resource usage.

C. End-to-End Delay

Fig. 2(b) shows the cumulative distribution of the end-to-end delay of the flow with 64 bytes flow length. N represents the number of flows. From the results, we can see that the delay of all flows is less than 100 μs. At N=100 and B=1 Gbps, the maximum end-to-end delay is about 48 μs, the average delay is about 33 μs, and the end-to-end delay of half of the flows is less than 35 μs. The average delay is approximate to the median of the delay, and the CDF curve is approximately linear, indicating that the end-to-end delay distribution is approximately uniform. This is because the same flow length makes the same serialization delay among flows. The CDF curve shows the same characteristics at B=10 Gbps. At N=100, the maximum end-to-end delay is about 12 μs, which is 1/4 of that at B=1 Gbps. At N=200, the maximum end-to-end delay is about 15 μs, which is about 1/6 of that at B=1 Gbps. The difference in the ratio is because the proportion of queuing delay in end-to-end delay increases with the number of flows.

D. Runtime

Fig. 2(c) shows the runtime of the scheduling algorithm with and without virtual queues. The schedule is computed with the ILP solver, SMT solver, and Tabu Search (TS) algorithm, respectively. The number of physical queues allocated to the ST flows is 1. The flows are all with 64 bytes length and a deadline of 1 ms. All flows are schedulable. The results show that the VQ-TAS scheme introduces few additional time cost with the SMT solver and TS algorithm. With the ILP solver, the runtime with virtual queues or not is approximately the same. From the view of algorithm operation, the VQ-TAS scheme only adds the sequence process and does not increase the constraints for the scheduling problem. The time cost of the sequence process is much smaller than that of the GCL computation algorithm.

V. Conclusion

This paper proposes a VQ-TAS scheme to schedule a large number of ST flows. In the scheme, the virtual queues are introduced to isolates each flow with a time offset parameter. The virtual queues are generated following the flows set without the limitation of the number of physical queues. The TAS problem is modeled as an optimization problem and a two-stage scheduling algorithm is designed. The proposed scheme is evaluated with a use case of the industrial automation. The simulation results show that the VQ-TAS scheme can schedule hundreds to thousands of flows while provide the end-to-end delay guarantees. Compared with the typical scheduling algorithms, the time cost of VQ-TAS scheme is almost the same.

Acknowledgment

This work is supported in part by the National Natural Science Foundation of China (Grant No.92067102), and in part by the project of Beijing Laboratory of Advanced Information Networks.

References

[1] W. Steiner, S. S. Craciunas, and R. S. Oliver, "Traffic planning for time-sensitive communication," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 42–47, Jul. 2018.

[2] S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner, "Scheduling real-time communication in ieee 802.1 qbv time sensitive networks," in *Proc. 24th ACM Int. Conf. Real-Time Netw. Syst.(RTNS)*, Brest, France, Oct. 2016, pp. 183–192.

[3] F. Dürr and N. G. Nayak, "No-wait packet scheduling for IEEE time-sensitive networks (TSN)," in *Proc. 24th ACM Int. Conf. Real-Time Netw. Syst.(RTNS)*, Brest, France, Oct. 2016, pp. 203–212.

[4] V. Gavrilut, L. Zhao, M. L. Raagaard, and P. Pop, "AVB-aware routing and scheduling of time-triggered traffic for TSN," *IEEE Access*, vol. 6, pp. 75 229–75 243, Jun. 2018.

[5] B. Briscoe, A. Brunstrom, A. Petlund, D. Hayes, D. Ros, J. Tsang, S. Gjessing, G. Fairhurst, C. Griwodz, and M. Welzl, "Reducing internet latency: A survey of techniques and their merits," *IEEE Commun. Surveys Tut.*, vol. 18, no. 3, pp. 2149–2196, 3rd Quart. 2014.

[6] "Industrial use cases iec/ieee 60802," IEC/IEEE, Tech. Rep., 2018.

[7] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. Elbakoury, "Performance comparison of IEEE 802.1 TSN time aware shaper (TAS) and asynchronous traffic shaper (ATS)," *IEEE Access*, vol. 7, pp. 44 165–44 181, Jul. 2019.