Enabling Low Latency for ECQF based Flow Aggregation Scheduling in Time-Sensitive Networking

Ping Liu[1], Tong Zhang[3], Xiaoqin Feng[1], Yanying Ma[1], Fengyuan Ren [1,2*]
Lanzhou University[1], Tsinghua University[2], Nanjing University of Aeronautics and Astronautics[3]

ABSTRACT

Cyclic Queuing and Forwarding (CQF) configures the same cycle length on the flow path, resulting in certain flows unschedulable. Enhanced CQF (ECQF) based flow aggregation utilizes variable cycle length to address this issue. However, it remains a conceptual model without a concrete implementation. In this paper, we propose a jointly optimize aggregation cycle and flows' offsets (JACO) mechanism to achieve ECQF-based flow aggregation. We also design an incremental heuristic algorithm for JACO. Finally, we evaluate the performance of JACO in different scenarios using OMNet++ simulation platform. Compared with ECQF, the results show that JACO reduces latency and improves resource utilization.

KEYWORDS

Traffic Sheduling, Time-Sensitive Networking (TSN), Enhanced Cyclic Queuing and Forwarding (ECQF), Flow aggregation

ACM Reference Format:
Ping Liu[1], Tong Zhang[3], Xiaoqin Feng[1], Yanying Ma[1], Fengyuan Ren [1,2]. 2024. Enabling Low Latency for ECQF based Flow Aggregation Scheduling in Time-Sensitive Networking. In *61st ACM/IEEE Design Automation Conference (DAC '24), June 23–27, 2024, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3649329.3655941

1 INTRODUCTION

Deterministic Real-time transmission is essential in time-sensitive application domains, such as autonomous driving, smart factories, and remote healthcare [1, 4, 6]. Time-Sensitive Networking (TSN) task group [16] has proposed a series of Ethernet enhancement standards to guarantee bounded end-to-end delay and reliability. Among these standards, traffic scheduling and shaping mechanisms account for a most important part [10].

IEEE 802.1Qch recommends the Cyclic Queuing and Forwarding (CQF) mechanism based on time synchronization [13]. CQF cyclically opens or closes the gate of two queues to transmit flows in the cycle length T_s. Compared to the Time-Aware Shaper (TAS) proposed by IEEE 802.1Qbv [12], CQF greatly simplifies the configuration of the Gate Control List (GCL) [17] and provides an easily calculable end-to-end delay. By selecting an appropriate cycle length T_s, CQF can guarantee the end-to-end latency for scheduled traffic (ST) is bounded.

CQF schedule ST flows based on fixed cycle length. However, the fixed scheduling cycle length configuration for each hop on the network path may cause some ST flows to exceed their deadlines of transmission. Moreover, determining the value of T_s is difficult because a too long T_s would increase the end-to-end latency of the flows, while a too short T_s would limit the number of scheduled flows.

In this case, Enhanced CQF (ECQF) based flow aggregation [3, 5] improves the schedulability of ST flows by adopting variable scheduling cycle length. ECQF [15] is based on frequency synchronization that divides the output queue into fixed-sized bins and cyclically output frames in these bins. Flow aggregation aggregates frames from one or multiple ST flows into the same bin, as if they were transmitted continuously. In other words, the ECQF-based flow aggregation aggregates flows at the first switch on the flow path and then cyclically outputs them in the aggregation cycle length along the remaining switches on the path. This mechanism can reduce end-to-end latency for ST flows and improve resource utilization by decreasing the length of the aggregation cycle.

However, ECQF-based flow aggregation is only defined as a model. It is essential to make it practically applicable. In this paper, we propose a mechanism that jointly optimizes aggregation cycle length and flows' offsets (JACO) to implement ECQF-based flow aggregation scheduling. We formalize the scheduling problem into a mixed integer linear programming problem. On this basis, an incremental heuristic algorithm is designed to compute the optimal aggregation cycle length and flows' offsets. We conduct extensive simulations to evaluate performance of JACO on the OMNet++ platform [11]. The results indicate that JACO mechanism can reduce end-to-end latency and improve resource utilization compared to the ECQF.

In general, the main contributions of this paper are as follows.

- Proposing JACO mechanism to achieve ECQF-based flow aggregation, which reduces the end-to-end latency of ST flows.
- Designing an incremental heuristic algorithm to search for the minimum aggregation cycle length and the corresponding flows' offsets.
- Conducting extensive simulation experiments to evaluate the performance of JACO under different scenarios.

The rest of the paper is organized as follows. We introduce the background in Section II and the motivation in Section III. Section IV describes the formulation of the flow aggregation scheduling problem. The design of the algorithm is presented in Section V. Section VI shows and analyzes the simulation results. The related work and the conclusion are presented in Section VII and Sections VIII.

*Corresponding author: renfy@tsinghua.edu.cn

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM ISBN 979-8-4007-0601-1/24/06…$15.00
https://doi.org/10.1145/3649329.3655941

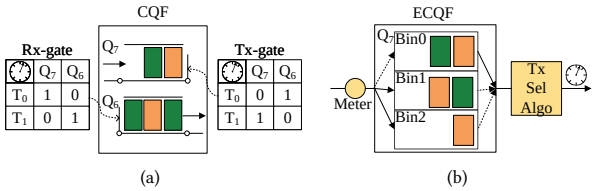


Figure 1: Switch models in CQF and ECQF.

2 BACKGROUND

In this section, we provide a detailed description of the frame forwarding rules in switch for CQF, ECQF, and ECQF-based flow aggregation.

As shown in Figure 1(a), the implementation of CQF requires two queues-queues 7 and 6. CQF cyclically changes each queue between queuing and forwarding states using Rx-gate and Tx-gate GCL. CQF divides the global time into equal time intervals of length T_s. In odd-numbered time intervals, queue 7 only receives incoming frames from its upstream node without forwarding them. Meanwhile, queue 6 only forwards frames received during the previous even-numbered time intervals to its downstream node and does not receive frames. In even-numbered time slots, queue 7 and queue 6 perform opposite operations. The boundary between the maximum and the minimum delay of the CQF is denoted by $L_{max} = (H+1)\times T_s$ and $L_{min} = (H-1)\times T_s$, respectively, where H is the number of hops along the ST flow path.

CQF configures the same scheduling cycle length T_s for each switch in the network, which limits the scheduling of some ST flows. In addition, determining the value of T_s is difficult. To address these problems, ECQF-based flow aggregation uses different scheduling cycle length for the first switch and other switches on the ST flow path.

As shown in Figure 1(b), frames of queue 7 are transmitted based on ECQF. ECQF divides queue 7 into fixed-size bins, such as Bin0, Bin1, and Bin2. These bins refer to consecutive and equally sized time intervals. We can set different bin sizes and bin numbers in each output queue of the switch as needed. These bins are output in a circular fashion at a constant frequency. The Meter in Figure 1(b) is a shaper that calculates the frames' dequeuing time based on their arrival time. The dequeue time is also called the eligibility time. The Tx Se Algo is a transmission selection algorithm that dequeues a frame when its eligibility time in the bin is less than or equal to the current system simulation time. Other queues can also be configured in the same way, but bin sizes must maintain an integer relationship [15].

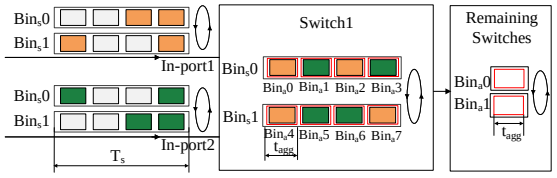


Figure 2: Switch model in ECQF-based flow aggregation.

Flow aggregation is based on the ECQF mechanism. It aggregates ST flows into the first switch on the flow path and outputs them with an aggregation cycle length t_{agg} along the remaining switches on the path. As presented in Figure 2, ST flows enter the first switch of their path from input ports 1 and 2, then they are aggregated into Bin_s0 and Bin_s1. ST flows aggregated within the time interval from 0 to T_s are output from the first switch at time T_s. On the remaining switches, they are cyclically output in bins of size t_{agg}, namely Bin_a0 and Bin_a1.

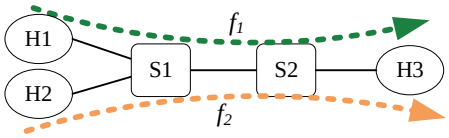


Figure 3: Network topology with two flows.

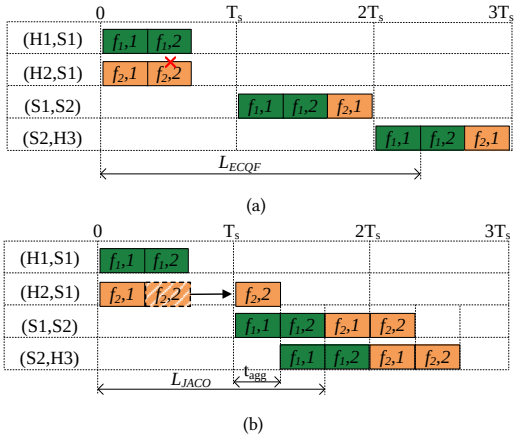


Figure 4: Scheduling Gantt chart with ECQF-based and ECQF-based flow aggregation mechanism. The rectangle f_i, j means jth frame of f_i.

3 MOTIVATION

In this section, we illustrate the motivation of the proposed JACO mechanism through an example.

Figure 3 presents the network topology. Flows f_1 and f_2 share identical flow characteristics. Their respective periods, deadlines, and frame counts are $4T_s$, $4T_s$, and 2. Figure 4 shows the Gantt chart for f_1 and f_2 in the topology based on different scheduling mechanism.

We map flows f_1, f_2 to the highest priority queue and regulate them to leave from the same egress port of the switch in the example. The queue of corresponding egress port in $S1$ and $S2$ is operated based on ECQF as shown in the Figure 4(a). The scheduling cycle length is T_s and flows offsets of f_1 and f_2 are 0. Frames $f_1.1$, $f_1.2$, $f_2.1$, and $f_2.2$ arrive at switch $S1$ between 0 and T_s, and their eligibility time is set to T_s. Frame $f_2.2$ will be dropped due to the limitation of the queue size. The latency of $f_1.1$ from source to destination according to ECQF is L_{ECQF}. However, if the queue of egress port in switch $S2$ is operated based on flow aggregation as shown in Figure 4(b). Frames $f_1.1$, $f_1.2$, and $f_2.1$ are aggregated in the switch $S1$ within the T_s and their eligibility time are T_s. Frame $f_2.2$ is delayed to next slot and its eligibility time is $2T_s$. Then, all frames are output in the switch $S2$ with aggregation cycle length t_{agg}, where $T_s = 3t_{agg}$ in the example. The latency of $f_1.1$ is L_{JACO}. The result indicates that L_{ECQF} is $\frac{2T_s}{3}$ greater than L_{JACO}.

Table 1: Notation

Symbol	Meaning
$G = V, E$	Network topology
V	Set of network nodes
E	Set of network links
$F = (f_1, ..., f_i)$	Set of period flows
$f_i.s$	Source of period flow i
$f_i.d$	Destination of period flow i
$f_i.\tilde{d}$	Deadline of period flow i
$f_i.T$	Period of period flow i
$f_i.\varphi$	Offset of period flow i
T_t	Transmission delay of a frame
T_p	Link propagation delay
T_o	Switch processing delay
δ	Phase difference
HP	Hyperperiod
T_s	Scheduled cycle length for period flows
Q	Queue size

In conclusion, the latency of ST flows can be reduced by minimizing aggregation cycle length and planing flows' offsets, as shown in the above example. Based on this, we propose the JACO mechanism that jointly optimizes aggregation cycle length as well as flows' offsets.

4 FORMULATION

In this section, we introduce system model that includes network model and traffic model. Then, we formulate the implementation of JACO. The symbols used in this paper are shown in Table 1. The time unit is microsecond.

4.1 System Model

Network Model. We present the network as a directed graph $G = \{V, E\}$, where $V = S \cup H$ denotes the set of switches and end devices in TSN network. The edge set E denotes the physical full-duplex link connecting the S and H nodes in the network. End devices can be of several types, such as sensor and electronic control unit (ECU). Typically, a sensor can be a camera or radar, and an ECU is composed of a CPU, memory, and network card. Each device can be both the source and the destination for sending and receiving data frames. Each switch forwards frames to the next hop based on the destination address. The link between v_a and v_b corresponds to two distinct unidirectional links, namely $[v_a, v_b]$ and $[v_b, v_a]$.

ECQF-based flow aggregation scheduling requires frequency synchronization, which is the process of ensuring that different clocks or signals within a system maintain a consistent frequency. In this paper, we assume all nodes clock are frequency synchronized and define the maximum phase difference between any two adjacent nodes is δ.

Traffic Model. In TSN, ST flows are responsible for carrying out periodic critical communication tasks. The application of H sends a flow in each period. We use f_i to denote a flow and F to denote the flow set. In this paper, each ST flow contains a frame. The maximum frame size is the Ethernet Maximum Transmission Unit (MTU) of 1500 bytes. A flow is characterized by a seven-tuple that includes the source, destination, deadline, period, offset, arrival time, and eligibility time, as shown in Eq.1. The arrival time t_a is the time when a frame arrives at the switch, and the eligibility time t_e is the time when the frame can depart from the switch.

$$\begin{aligned} &\forall f_i \in F, i \in [0, n-1], \\ &f_i = \left\{ f_i.s, f_i.d, f_i.\tilde{d}, f_i.T, f_i.\varphi, f_i.t_a, f_i.t_e \right\} \end{aligned} \tag{1}$$

In the process of flow scheduling, it is essential to plan and schedule flows within a hyperperiod HP, which is the least common multiple of all flow periods as defined in Eq.2. Therefore, within a hyperperiod, each flow may occur more than once.

$$HP = LCM(F.T) \tag{2}$$

4.2 Problem Formalization

Based on above system model, we formalize the ECQF-based flow aggregation scheduling problem into a mixed integer linear programming problem. The decision variables are the aggregation cycle length and flows' offsets, whose optimal value can be calculated by solving the problem. The constraints and objective are elaborated as follows.

Offset Constraint. To ensure that all flows are transmitted in order according to their generation time, frames need to be sent within their own periods and should not conflict with those from future periods. This is because if a frame is not sent within its period, frames from multiple periods can accumulate at the end device, increasing the end-to-end latency. The offset constraint is presented in Eq. 3. Additionally, the constraint confines the search space within a reasonable range and reduces the complexity of the search.

$$\begin{aligned} &\forall f_i \in F, i \in [0, n-1], \\ &0 \le f_i.\varphi < \frac{f_i.T}{T_s} \end{aligned} \tag{3}$$

Cycle Length Constraint. This constraint defines the upper and lower bounds of the cycle length. The offset should be an integral multiple of T_s, so the upper bound of the cycle length is the greatest common divisor (GCD) of all flow periods as defined in Eq. 4.

$$max(T_s) = GCD(F.T) \tag{4}$$

The lower bound of cycle length needs to ensure that all frames in the same queue can be forwarded, which is described in Eq. 5,

$$min(T_s) = \frac{Q \times MTU}{B} + T_o + T_p + \delta \tag{5}$$

where Q is the maximum frame number that the queue can hold, B is the link bandwidth, T_p is the link propagation delay, T_o is the switch processing delay and δ is the maximum phase difference.

Aggregation Cycle Constraint. This constraint limits the maximum and minimum aggregation cycle length as in Eq. 6. To prevent frame loss, the number of buffered frames within each aggregation cycle length must not exceed the permitted transmission limit. The maximum value should be T_s. This is because when the t_{agg} equals T_s, the scheduling result is same as that based on ECQF. Therefore, t_{agg} should not exceed T_s to achieve latency reduction. The minimum aggregation cycle length t_d should guarantee that an MTU is forwarded within the cycle.

$$
\begin{aligned}
&\forall f_i \in F, i \in [0, n-1], (v_a, v_x), (v_x, v_b) \in E, \\
&t_d \leq t_{\text{agg}} \leq T_{\text{s}}, \\
&f_i^{(v_x,v_b)}.t_{\text{a}} = f_i^{(v_a,v_x)}.t_{\text{e}} + (k-1) \times t_d + L_d, \\
&f_i^{(v_x,v_b)}.t_{\text{e}} = (f_i.\varphi + 1) \times T_s + \\
&(floor \frac{f_i^{(v_x,v_b)}.t_a - (f_i.\varphi+1) \times T_s}{t_{agg}} + 1) \times t_{agg}, \\
&count(f_i^{(v_x,v_b)}.t_{\text{e}}) \leq \frac{t_{agg}}{t_d}
\end{aligned} \tag{6}
$$

where L_d is the transmission latency of actual ST frames in the network, and it is composed of frame transmission delay T_t, link propagation delay T_p, and switch processing delay T_o, i.e., $L_d = T_t + T_p + T_o$. For example, if the bandwidth of all links is 1Gbps, then T_t of a frame with a size 1000 bytes is 8.336μs. When T_p and T_o are both 1μs, $L_d = T_t + T_p + T_o = 8.336 + 1 + 1 = 10.336\mu$s. $f_i^{(v_a,v_x)}.t_{\text{e}}$ is the eligibility time of the frame at the switch v_a, k is the kth frame with eligibility time $f_i^{(v_a,v_x)}.t_{\text{e}}$, $f_i^{(v_x,v_b)}.t_{\text{a}}$ is the arrival time of the frame at switch v_x, and $f_i^{(v_x,v_b)}.t_{\text{e}}$ is the eligibility time of the frame at the switch v_x. The $floor$ is the downward rounding function and *count* is the counting function.

Deadline Constraint. This constraint ensures that the latency of flows is restricted within their deadlines. As shown in Figure 2, flows are aggregated in the first switch with T_s and then output them with aggregation cycle length t_{agg}. The constraint is illustrated in Eq.7.

$$
\begin{aligned}
&\forall f_i \in F, i \in [0, n-1], \\
&(f_i.\varphi + 1) \times T_{\text{s}} + (H_i - 1) \times t_{agg} \leq f_i \cdot \tilde{d}
\end{aligned} \tag{7}
$$

where H_i is the number of hops on the path of flow f_i and $f_i.\tilde{d}$ is the deadline of flow f_i.

Queue Size Constraint. The queue size is the maximum number of frame that a queue can buffer. When both frames requiring aggregation and frames scheduled based on aggregation cycle length coexist on the same switch, the total number of frames from both types must not exceed the queue size. The constraint is shown in Eq. 8.

$$
\begin{aligned}
&\forall f_i \in F, i \in [0, n-1], (v_1, v_2) \in E \\
&count(f_i^{(v_1,v_2)}.t_{\text{e}}) \leq \frac{Q \times MTU}{B} - \frac{t_{agg}}{L_d}
\end{aligned} \tag{8}
$$

where $f_i^{v_1,v_2}.t_{\text{e}}$ refers to the eligibility time of the frame that enters the first switch v_1 on its path.

Objective. The optimization objective is to minimize the end-to-end latency of all flows by minimizing aggregation cycle length and planing flows' offsets. The objective is represented as Eq.9:

$$
minimize \sum_{i=0}^{n-1} ((f_i.\varphi + 1) \times T_{\text{s}} + (H_i - 1) \times t_{agg}) \tag{9}
$$

5 ALGORITHM DESIGN

In this section, we introduce an incremental heuristic scheduling algorithm to solve the optimization problem in the previous section. Firstly, we set the aggregation cycle length to the minimum value t_d. Secondly, we incrementally plan the offset of each frame to satisfy all transmission constraints. The aggregation cycle length is incremented by t_d, and the offset planing process is repeated using the new cycle length value until a combination of aggregation cycle length and corresponding flows' offsets is found. Lastly, the algorithm returns t_{agg} and the corresponding offset of each flow.

Algorithm 1: Heuristic algorithm

```
   Data: F, Network topology G
   Result: t_agg, F.φ
1  Initialize T_s, HP, F.φ, Q
2  for t_agg ∈ [t_d, t_d, T_s] do
3      for f_i ∈ F do
4          for f_i.φ ∈ [0, f_i.T/T_s) do
5              delay_e2e = constraint_e2e(f_i, t_agg)
6              if delay_e2e ≤ f_i.deadline then
7                  for node ∈ f_i.route do
8                      t_elg = computeEligibilityTime(f_i, t_agg)
9                  Q_length =
                     computeQueueLength(f_i, t_agg, t_elg)
10                 t_agg_count = countNum(t_agg, t_elg)
11                 if Q_length ≤ Q then
12                     if t_agg_count ≤ t_agg/L_d then
13                         F.φ_arr = f_i.φ
14                         break
15     return min(t_agg), F.φ
```

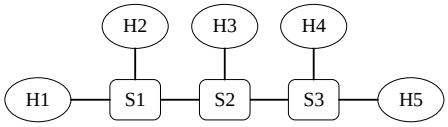


Figure 5: Network topology.

Algorithm 1 shows the detailed heuristic algorithm. The input consists of the parameters of ST flows and the network topology. The output results are the minimum aggregation cycle length t_{agg} and corresponding flows' offsets $F.\varphi$. In line 1, the algorithm initializes scheduled cycle length T_s, hyperperiod HP, flows' offsets $F.\varphi$ and queue size Q. The algorithm incrementally searches for the offsets of the flows to satisfy the defined constraints in lines 3-14 when t_{agg} takes a value. The offset constraint is met in line 4 by Eq.3, and Eq.7 is used to determine whether the end-to-end latency meets the constraint in line 5. Lines 7-12 check whether the queue size constraint and aggregation cycle length constraint are met Eqs.8 and 6 by counting the number of frames with the same eligibility time. If the offsets cannot meet all constraints at the value of t_{agg}, the algorithm increases t_{agg} by t_d in line 2 and performs a new offset search. Finally, the algorithm returns the satisfactory t_{agg} and all flows' offsets $F.\varphi$.

6 EVALUATION

In this section, we evaluate the performance of JACO mechanism in both a benchmark model and an industrial scenario. Our simulations are conducted on the OMNeT++ platform [11] with the INET4.4.1 framework [8].

6.1 Benchmark Model

Simulation Setup. The simulation network topology is shown in Figure 5, and flow features are shown in Table 2. We set the link

Table 2: Flow Features

Flow	Src.	Dest.	Period(μs)	Size(B)	Deadline(μs)	Type
f_1	H1	H5	300	100	300	ST
f_2	H2	H5	450	100	450	ST
f_3	H3	H5	600	100	600	ST
f_4	H4	H5	900	100	900	ST

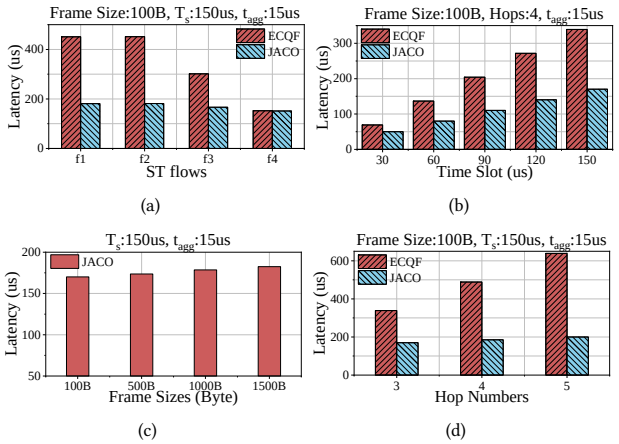


Figure 6: End-to-end latency. (a) with different mechanisms. (b) with different T_s. (c) with different frame sizes. (d) with different hops.

bandwidth to 1Gbps. Due to the limitations in end-to-end latency and the on-chip memory of the switch, frames cannot be buffered within the switch for a long time. Therefore, we set the queue size to 10 (the queue can buffer up to 10 MTU-sized frames) and the scheduling cycle length to 150μs. The clocks in the simulation are set to be frequency synchronized and the range of phase difference is between 0.3μs and 1μs. All ST flows are mapped to the highest priority queue in the simulation.

Simulation Results. We design a performance comparison between JACO-based and ECQF-based scheduling under different parameters. T_s is set to 150μs for ECQF scheduling. For scheduling mechanism JACO, the settings are configured as follows: T_s is set to 150μs, t_{agg} is set to 15μs, and the flows offset are set to 0. The results of end-to-end latency and resource usage are shown in Figure 6 and Figure 7.

The comparison of end-to-end latency under different parameter configurations is shown in Figure 6. We present the latency of four flows based on JACO and ECQF. The latency using JACO is lower than that using ECQF in Figure 6(a). This is because the former schedules ST flows using the scheduled cycle length T_s only at the first hop and using aggregation cycle length t_{agg} at the remaining switches, whereas the latter schedules ST flows using the scheduled cycle length T_s at every hop. The more hops a flow traverses, the greater latency difference between the two scheduling mechanisms is. Figure 6(b) shows the comparison of the average latency of all ST flows when T_s takes different values. As T_s increases, ECQF exhibits faster latency growth than JACO because it increments at each hop, whereas JACO increments only at the first hop. Figure6(c) presents that the average latency of all ST flows in JACO does not significantly increase with an increase in frame sizes. This is because the delay is only influenced by the frame size at the last hop, whereas it remains fixed latency at other switches. In the last Figure 6(d), as the number of hops increases, the latency in ECQF grows by a multiple of T_s, whereas the latency in JACO grows by a multiple of t_{agg}. Consequently, the growth in ECQF is significantly greater than that in flow aggregation.

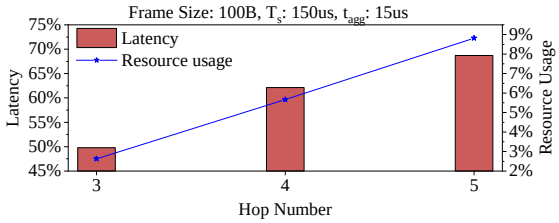


Figure 7: Latency and resource usage.

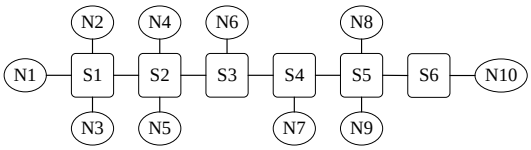


Figure 8: Industry network topology.

Figure7 illustrates the variations in average end-to-end latency and average resource utilization based on JACO and ECQF as the network topology hop count changes. We can observe that as the number of hops increases for the flow, compared to ECQF, JACO's average latency is reduced by 50% at 3 hops, 62% at 4 hops, and 69% at 5 hops. Additionally, in terms of average resource utilization, there is an increase of 3% at 3 hops, 6% at 4 hops, and 9% at 5 hops.

6.2 Industrial Scenario

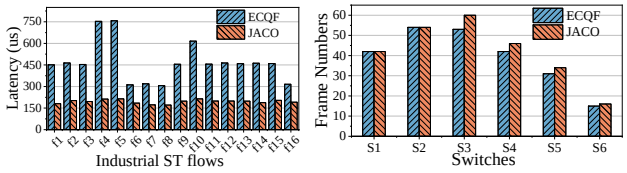
Simulation Setup. Figure 8 illustrates a typical industrial scenario consisting of ten end devices and six switches. The network link bandwidth is 1Gbps and switch queue size is 10. Table 3 describes the flow parameters. The network topology and flow characteristics follow IEC/IEEE 60802 [14]. To assess the performance of JACO in the industrial scenario, we configure sixteen ST flows to be mapped into a single queue with T_s=150μs. We conduct a simulation to compare the transmission performance between JACO and ECQF.

Simulation Results. Figure 9 shows the comparison of two mechanisms in end-to-end latency and resource utilization. Compared to ECQF, the proposed JACO results in lower latency in 9(a). Moreover, the latency of some flows based on ECQF exceeds their deadline, such as f_1, f_2, f_3, f_4, f_9, f_{10}, f_{11}, f_{14}, and f_{15}. This is because the scheduling cycle length within each hop based on ECQF is same. Whereas, JACO uses different cycle length between first switch and other switches along the path. 9(b) demonstrates the number of frames passing through each hop within the hyperperiod. Compared to ECQF, JACO transmits a greater number of frames to the destinations.

We experimentally evaluate end-to-end latency and resource usage in the industrial scenario. The results indicate that the proposed JACO reduces the end-to-end average latency by 58.3% and improves the average resource usage by 6% compared to ECQF.

Table 3: Industrial Flow Features

Flow	Src.	Dest.	Period(μs)	Size(B)	Deadline(μs)	Type
f_1	N1	N6	300	100	300	ST
f_2	N2	N7	300	500	300	ST
f_3	N3	N7	300	100	300	ST
f_4	N4	N10	450	300	450	ST
f_5	N5	N10	900	400	900	ST
f_6	N6	N9	450	600	450	ST
f_7	N8	N10	600	1000	600	ST
f_8	N9	N10	600	700	600	ST
f_9	N1	N7	300	100	300	ST
f_{10}	N3	N8	300	500	300	ST
f_{11}	N2	N6	300	500	300	ST
f_{12}	N5	N9	600	400	600	ST
f_{13}	N4	N8	600	300	600	ST
f_{14}	N7	N10	450	1000	450	ST
f_{15}	N3	N6	300	500	300	ST
f_{16}	N6	N9	900	600	900	ST



(a) (b)

Figure 9: Comparison of the two mechansims in the industrial scenario.

7 RELATED WORK

To ensure the deterministic transmission of ST flows, many researchers have investigated various scheduling mechanisms. CQF is considered a preferable choice among them, owing to its simple GCL configuration. The CQF mechanism can be classified into two types: two-queue CQF scheduling [7, 18–20], and the extension of CQF with three or more queues [2, 9]. The injection time planning (ITP) mechanism was proposed in [19] which plans offsets for all ST flows and improves the number of scheduled flows and resource utilization. [20] introduces an incremental scheduling algorithm with flow judgment conditions and based on position diversity variable search bound scheduling algorithm. [7] designs a scheduling mechanism based on mapping scores between flow ordering and offset. Both [20] and [7] reduce the run time for computing solution and ensure load balancing. By jointly computing routing path and planning offset, [18] enhances the success rate of ST flow scheduling. The extended CQF mechanisms [2, 9] utilize three or more queues and a cycle identifier to process frames that arrive in the error cycle. In summary, existing works are based on the scheduling of fixed cycle length. In this paper, ECQF-based flow aggregation adopts different scheduling cycle lengths to reduce latency.

8 CONCLUSION

In this paper, we propose a mechanism named JACO which jointly optimizes aggregation cycle length and flows' offsets (JACO) to achieve ECQF-based flow aggregation scheduling. We formalize the ECQF-based flow aggregation scheduling problem into a mixed integer linear programming problem. On this basis, we design an incremental heuristic algorithm to minimize end-to-end latency by searching for the optimal aggregation cycle length and flows' offsets. Finally, the transmission performance in two different scenarios is evaluated using extensive simulations on the OMNet++ simulator. The results show that JACO significantly reduces the end-to-end latency and improves the resource utilization when compared with ECQF.

9 ACKNOWLEDGE

We thank the anonymous reviewers for their valuable comments. This work is supported in part by the National Key Research and Development Program of China (No. 2022YFB2901404), and by National Natural Science Foundation of China (NSFC) under Grant No. 62132007, 62221003. Correspondence to: Fengyuan Ren (renfy@tsinghua.edu.cn), Tong Zhang (zhangt@nuaa.edu.cn).

REFERENCES

[1] Sébastien Bigo, Nihel Benzaoui, Konstantinos Christodoulopoulos, Ray Miller, Wolfram Lautenschlaeger, and Florian Frick. 2021. Dynamic Deterministic Digital Infrastructure for Time-Sensitive Applications in Factory Floors. *IEEE Journal of Selected Topics in Quantum Electronics* 27, 6 (2021), 1–14. https://doi.org/10.1109/JSTQE.2021.3093281

[2] Mach Chen, Xuesong Geng, and Zhengiang Li. 2019. Segment Routing (SR) Based Bounded Latency. https://datatracker.ietf.org/doc/html/draft-chen-detnet-sr-based-bounded-latency-01 Internet Engineering Task Force.

[3] Shuang Chen and Tongtong Wang. 2022. Considerations on stream aggregation in enhanced CQF scheme. https://www.ieee802.org/1/files/public/docs2022/dv-SChen-StreamAggregationEnhencedCQF-0916-v01.pdf

[4] Libing Deng, Guoqi Xie, Hong Liu, Yunbo Han, Renfa Li, and Keqin Li. 2022. A Survey of Real-Time Ethernet Modeling and Design Methodologies: From AVB to TSN. *ACM Comput. Surv.* 2 (2022), 1–36. https://doi.org/10.1145/3487330

[5] Norman Finn. 2022. Enhanced CQF and Stream Aggregation. https://www.ieee802.org/1/files/public/docs2022/dv-finn-CQF-stream-aggregation-v01.pdf

[6] Bernard Fong, A. C. M. Fong, and Kim-Fung Tsang. 2021. Capacity and Link Budget Management for Low-Altitude Telemedicine Drone Network Design and Implementation. *IEEE Communications Standards Magazine* 5, 4 (2021), 74–78. https://doi.org/10.1109/MCOMSTD.0001.2100010

[7] Miao Guo, Chaojie Gu, Shibo He, Zhiguo Shi, and Jiming Chen. 2023. MSS: Exploiting Mapping Score for CQF Start Time Planning in Time-Sensitive Networking. *IEEE Transactions on Industrial Informatics* 19, 2 (2023), 2140–2150. https://doi.org/10.1109/TII.2022.3206815

[8] INET 2022. INET Framework. Version 4.4.1. https://inet.omnetpp.org/2022-07-27-INET-4.4.1-released.html

[9] Qiang Li, Xuesong Geng, Bingyang Liu, Eckert Toerless, and Liang Geng Geng. 2019. Large-Scale Deterministic IP Network. https://datatracker.ietf.org/doc/html/draft-qiang-detnet-large-scale-detnet-04/ Internet Engineering Task Force.

[10] John L. Messenger. 2018. Time-Sensitive Networking: An Introduction. *IEEE Communications Standards Magazine* 2, 2 (2018), 29–33. https://doi.org/10.1109/MCOMSTD.2018.1700047

[11] OMNet++ 2022. OMNet++ Simulation. Version 6.0.1. https://omnetpp.org/

[12] Standard 2015. IEEE 802.1Qbv Standard. https://www.ieee802.org/1/pages/802.1bv.html

[13] Standard 2017. IEEE 802.1Qch Standard. https://1.ieee802.org/tsn/802-1qch/

[14] Standard 2018. IEC/IEEE 60802 TSN Profile for Industrial Automation. https://1.ieee802.org/tsn/iec-ieee-60802/

[15] Standard 2023. Amendment: Enhancements to Cyclic Queuing and Forwarding. https://www.ieee802.org/1/files/public/docs2023/dv-finn-proposed-text-0123-v02.pdf

[16] TSN 2012. IEEE Time-Sensitive Networking Task Group. https://www.ieee802.org/1/pages/tsn.html

[17] Marek Vlk, Zdeněk Hanzálek, Kateřina Brejchová, Siyu Tang, Sushmit Bhattacharjee, and Songwei Fu. 2020. Enhancing Schedulability and Throughput of Time-Triggered Traffic in IEEE 802.1Qbv Time-Sensitive Networks. *IEEE Transactions on Communications* 68, 11 (2020), 7023–7038. https://doi.org/10.1109/TCOMM.2020.3014105

[18] Xiaolong Wang, Haipeng Yao, Tianle Mai, Zehui Xiong, Fu Wang, and Yunjie Liu. 2023. Joint Routing and Scheduling With Cyclic Queuing and Forwarding for Time-Sensitive Networks. *IEEE Transactions on Vehicular Technology* 72, 3 (2023), 3793–3804. https://doi.org/10.1109/TVT.2022.3216958

[19] Jinli Yan, Wei Quan, Xuyan Jiang, and Zhigang Sun. 2020. Injection Time Planning: Making CQF Practical in Time-Sensitive Networking. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*. IEEE Press, 616–625. https://doi.org/10.1109/INFOCOM41043.2020.9155434

[20] Yanzhou Zhang, Qimin Xu, Lei Xu, Cailian Chen, and Xinping Guan. 2022. Efficient Flow Scheduling for Industrial Time-Sensitive Networking: A Divisibility Theory-Based Method. *IEEE Transactions on Industrial Informatics* 18, 12 (2022), 9312–9323. https://doi.org/10.1109/TII.2022.3151810