Joint Algorithm of Message Fragmentation and No-Wait Scheduling for Time-Sensitive Networks

Xi Jin*†‡, *Member, IEEE*, Changqing Xia*†‡, *Member, IEEE*, Nan Guan§, and Peng Zeng*†‡
*Key Laboratory of Networked Control Systems, Chinese Academy of Sciences, Shenyang 110016, China
†Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China
‡Institutes for Robotics and Intelligent Manufacturing, Chinese Academy of Sciences, Shenyang 110169, China
§Department of Computing, Hong Kong Polytechnic University, Hong Kong

***Abstract*—Time-sensitive networks (TSNs) support not only traditional best-effort communications but also deterministic communications, which send each packet at a deterministic time so that the data transmissions of networked control systems can be precisely scheduled to guarantee hard real-time constraints. No-wait scheduling is suitable for such TSNs and generates the schedules of deterministic communications with the minimal network resources so that all of the remaining resources can be used to improve the throughput of best-effort communications. However, due to inappropriate message fragmentation, the real-time performance of no-wait scheduling algorithms is reduced. Therefore, in this paper, joint algorithms of message fragmentation and no-wait scheduling are proposed. First, a specification for the joint problem based on optimization modulo theories is proposed so that off-the-shelf solvers can be used to find optimal solutions. Second, to improve the scalability of our algorithm, the worst-case delay of messages is analyzed, and then, based on the analysis, a heuristic algorithm is proposed to construct low-delay schedules. Finally, we conduct extensive test cases to evaluate our proposed algorithms. The evaluation results indicate that, compared to existing algorithms, the proposed joint algorithm improves schedulability by up to 50%.**

***Index Terms*—Time sensitive network, real-time scheduling, message fragmentation, networked control system**

I. Introduction

Time-sensitive networks (TSNs) are an emerging industrial network technology based on Ethernet networks and extend a set of IEEE standards to improve the controllability of industrial networks. Thus, in addition to best-effort communications supported by Ethernet networks, TSNs also support deterministic communications that have been widely considered as an effective solution to guarantee end-to-end delay constraints in hard real-time industrial systems.

In networked control systems, TSNs have been adopted [1], [2], [3]. Deterministic communications contain control commands and critical sensing data, and the other data adopts best-effort communications [4]. Deterministic communications and best-effort communications have different objectives. The real-time performance of deterministic communications is the most important. Before they start to transmit, their deterministic schedules must be generated so that the transmission process can be controlled to guarantee hard real-time constraints. For best-effort communications, there is no strict constraint. Their schedules do not need to be generated in advance, and networks just try to transmit them as soon as possible. In a TSN switch, the two kinds of communications share a fixed number of output queues. Deterministic communications must be assigned dedicated queues, while best-effort communications require more queues to improve network throughput [5]. Therefore, determining how to assign and utilize the queues are the key to improving network performance.

This paper focuses on store-and-forward switching, because compared to cut-through switching, store-and-forward switching is supported by more off-the-shelf TSN products. For example, CISCO IE 4000 [6] and NXP SJA1105 [7] support only store-and-forward switching, while no off-the-shelf TSN product supports only cut-through switching. In store-and-forward networks, no-wait scheduling is an effective method to make a performance trade-off between deterministic communications and best-effort communications [8]. Under no-wait scheduling, once a network switch receives a packet, it sends the packet immediately, i.e., only one queue is needed to cache the scheduled packets. Thus, when a no-wait scheduling algorithm is used to generate schedules for deterministic communications, except the occupied queue, all of the other queues can be used by best-effort communications. Therefore, no-wait scheduling algorithms not only generate schedules for deterministic communications on the dedicated queue, but also improve the performance of best-effort communications.

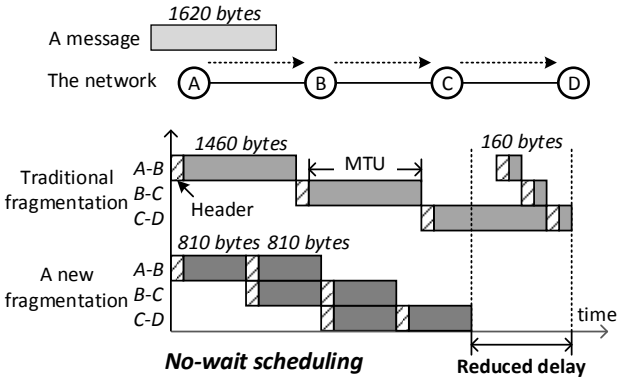


Fig. 1: Effect of message fragmentation on no-wait scheduling

However, when no-wait scheduling algorithms are adopted, the following issue should be considered. On the IP layer, a large message from the TCP layer must be fragmented into several smaller pieces. Each piece and its added packet header constitute a complete packet. These packets are then transmitted separately and reassembled at destination

devices. In this process, fragmentation algorithms will affect the transmission delay because smaller packets have higher parallelism. For example, in Fig. 1, a message of 1620 bytes must be fragmented into two pieces. In traditional Ethernet fragmentation, the length of the first packet is equal to the maximum segment size (MSS) plus a packet header, and the remaining part is contained in the second packet. Since no-wait scheduling algorithms do not allow a time interval between two consecutive hops of a packet, the second packet is postponed by the first packet and cannot be transmitted earlier. In another fragmentation, to reduce the long delay, the message can be fragmented into two pieces of 810 bytes each. Although the number of packets is unchanged, the delay is reduced by approximately one-third. Message fragmentation exists in TSNs [5]. However, there is currently no work that considers message fragmentation algorithms and no-wait scheduling algorithms together. This has led to some optimizable solution spaces being ignored.

Therefore, in this paper, a joint algorithm is proposed that fragments deterministic messages into packets of optimized sizes and schedules the packets under hard real-time constraints based on a no-wait strategy to improve the real-time performance of TSNs. When designing the joint algorithm, the following challenge is present. In the interest of saving resources, large packets should be used to reduce the overhead of fragmenting and reassembling. However, in the interest of temporality, a message should be fragmented into small packets, because small packets have higher parallelism and are easy to schedule. Thus, to address the above challenge, our proposed algorithms make a trade-off between these two aspects. The specific contributions are as follows.

First, a specification for the joint problem based on optimization modulo theories (OMT) is proposed. Since the joint problem is NP-hard, there is no polynomial time algorithm for finding optimal solutions. Thus, we formulate the problem as an OMT specification with real-time constraints to minimize the number of fragmented packets, and then invoke off-the-shelf solvers, e.g. Microsoft Z3, to solve it. Our evaluations indicate that, in an acceptable time, the OMT-based method can find optimal solutions for small networks.

Second, to improve the real-time performance of large networks, we calculate the worst-case delays of messages based on the recursive function shown in Theorem 2. Then, two corollaries are proposed on how to construct low-delay transmissions.

Third, based on the two corollaries, a joint algorithm is proposed that uses packets from large to small to construct schedules under hard real-time constraints. Thus, the proposed joint algorithm strikes a balance between saving resources and temporality. Extensive test cases were run to evaluate the joint algorithm. The evaluation results demonstrate that, compared to existing algorithms, the proposed joint algorithm improves schedulability by up to 50% and increases a small number of packets.

The rest of this paper is organized as follows: Section II reviews two categories of related work, including message fragmentation and real-time scheduling algorithms. Section III details the system model and problem. Section IV formulates the problem as an OMT specification. Section V proposes a heuristic algorithm to improve scalability of our algorithms. Section VI evaluates the proposed algorithms based on extensive test cases. Section VII concludes the paper.

II. Related Work

Much research has been proposed to improve the real-time performance of industrial networks, e.g., real-time scheduling [9], resource allocation [10] and data recovery [11]. In this paper, we only focus on the real-time scheduling problem of industrial TSNs. The work in [5] formulates the basic scheduling problem of TSNs as a specification, and then uses an off-the-shelf solver to calculate schedules. Then, the work in [12] introduces the capacity limitation of TSN switches into the basic problem so that the proposed scheduling algorithms can be adopted in actual systems. Based on the above work, the work in [13] proposes a loose scheduling algorithm to transmit massive data packets using limited switch resources. In addition to the classical scheduling problem, some extended problems have been considered. The work in [14] proposes an incremental scheduling algorithm to handle dynamic data flows. The work in [15] focuses on the runtime reconfiguration of TSNs and designs a heuristic algorithm to minimize the impact of reconfiguring switches on existing data flows. The work in [16] proposes a joint routing and scheduling algorithm to guarantee the real-time performance of time-triggered communications and reduce the end-to-end delays of audio-video-bridging (AVB) communications. The work in [17] also focuses on the joint problem of routing and scheduling and evaluates extensive test cases to explore the practical limitations of the integer linear programming for TSNs. The work in [18] evaluates asynchronous scheduling algorithms in TSNs and demonstrates that asynchronous scheduling algorithms are suitable for sporadic communications. The work in [19] analyzes the worst-case delays of AVB communications based on network calculus, and then the real-time performance of AVB communications can be precisely assessed. No-wait scheduling can significantly improve system efficiency and has been widely researched in many industrial systems, e.g. energy-efficient production planning systems [20], metal-processing systems [21] and flexible manufacturing systems [22]. In [8], no-wait scheduling is introduced into TSNs. However, the objective of the above work is to minimize the total length of generated schedules. In this paper, communications have different real-time requirements, and the above work cannot be adopted.

Message fragmentation is a widely used network technology. In Ethernet networks, to reduce the message response time, the work in [23], [24] optimizes and re-configures the maximum transmission unit size. The work in [25] designs a middleware for Ethernet networks to fragment large objects so that the worst-case delay of object-oriented transmissions can be reduced. The work in [26] evaluates four fragmentation methods based on Ethernet networks. In wireless local area networks, the work in [27], [28] designs novel fragmentation algorithms to improve the throughput of streaming videos. In the 6LoWPAN protocol, the work in [29], [30] demonstrates

that packet fragmentation is beneficial to energy consumption, throughput, and end-to-end delay. In delay tolerant networks, the work in [31] fragments large data items to multiple routing paths such that routing performance can be improved. In community antenna television networks, the work in [32] considers fragmentation and scheduling together. It first schedules real-time packets and then fragments best-effort packets to make them suitable for the free slots between successive real-time packets. In TTEthernet networks, a similar method is proposed to fragment event-triggered messages to make them schedulable in idle time intervals [33]. However, the above work cannot be applied to the multi-hop no-wait scheduling problem that is addressed in this paper, and there is no work to consider the effect of message fragmentation on no-wait scheduling. Therefore, in this paper, this effect will be analyzed, and joint algorithms will be proposed to improve network performance.

III. System Model and Problem Statement

In this section, we will detail our system model and problem. The symbols used in this paper are summarized in Appendix.

A TSN is characterized by a two-tuple $< N, L >$. The node set $N = \{n_1, n_2, ...\}$ includes TSN switches with multiple ports and end systems with only one port. The element $l_{i,j}$ in the link set L denotes a cable between nodes n_i and n_j. TSN switches form a mesh topology, and then end systems are connected to the TSN switches, as shown in Fig. 2. All ports and cables are full-duplex.

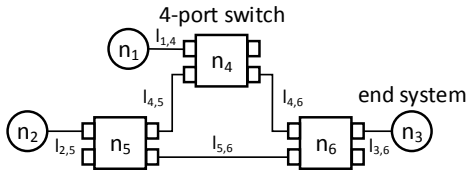


Fig. 2: A time sensitive network

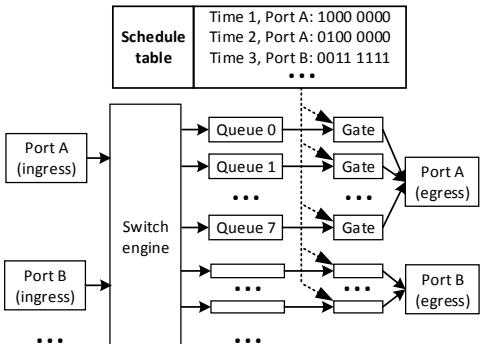


Fig. 3: Architecture of a TSN switch

In TSN switches, each egress port connects multiple queues, and each queue has a gate that can accurately control the sending of packets. Each TSN switch has a schedule table that records the precise times to open and close the gates. For example, as shown in Fig. 3, at time 1, the table entry that controls port A: queue 0 is opened, and the other queues are closed. The queue with a smaller ID has lower priority. When more than one gate is opening, the packet buffered in the higher-priority queue can occupy the egress port. As described in the IEEE 802.1bQu standard, the packet in the higher-priority queue is allowed to interrupt the process of sending the packet in the lower-priority queue, and the interrupted packet will be re-assembled in the next switch. In no-wait scheduling, deterministic packets are assigned to the highest-priority queue, and the queue is always opened such that the packets will not be blocked. Although blocking packets may reduce conflicts and improve schedulability, our work still adopts no-wait scheduling for two reasons. First, no-wait scheduling is not limited by the size of the schedule table. In switch chips, the size of the schedule table is limited and fixed. Thus, switches support a limited number of gate operations. Since no-wait scheduling algorithms do not operate gates, they do not use the schedule table and do not need to consider the table size constraint. Second, under no-wait scheduling, there is no schedule dissemination overhead. In a traditional network, a centralized scheduler disseminates the generated schedules to the corresponding network devices. However, under no-wait scheduling, switches forward the received packets immediately and do not need schedule information, so there is no schedule dissemination overhead. We ignore the other details of TSN switches since they have nothing to do with our proposed algorithms. Earlier works [13], [34] can help readers understand TSN switches in details.

Transmitting methods are designed herein only for deterministic flows. Best-effort flows are delivered by the classical best-effort services. In the following, if no specific description is given, flows refer to deterministic flows. A flow in the flow set F is characterized by a four-tuple $< p_i, d_i, \mathfrak{s}_i, \Pi_i >$, which denotes the period, relative deadline, message size, and routing path, respectively. Flow f_i generates a message of $\mathfrak{s}_i$ bytes periodically with a period p_i, and its relative deadline d_i is not greater than its period p_i. It is assumed that all flows generate their first messages at time 0. The jth message of flow f_i, called message $m_{i,j}$, is generated at time $j \times p_i$, and its absolute deadline is $j \times p_i + d_i$. The time interval $[j \times p_i, j \times p_i + d_i)$ is called its *active interval*. Only determining how to transmit data in a hyperperiod H that is defined as the least common multiple of all periods, i.e., $H = LCM\{p_1, p_2, ...\}$, is considered here, after the first hyperperiod, the subsequent hyperperiods are periodically repeated. The transmission time of a packet with x bytes is equal to $\frac{x}{v}$, where v denotes the transmission speed. The routing path Π_i is a link set, and $\Pi_i \subseteq L$. A path is from an end system to another end system. Hence, the number of hops, denoted r_i and $r_i = |\Pi_i|$, is not less than 2. Since routing is already well-studied, in the proposed model all routing paths are the shortest and generated based on existing algorithms, e.g., [35], [36].

From Fig. 1, it can be seen that message fragmentation based on the fixed MSS is not suitable for improving real-time performance. To address this problem, first, the sizes of deterministic packets are changed based on the requirements of no-wait scheduling, and then the packets are injected at a specified time that is determined by the proposed real-time

scheduling algorithm. However, the number of deterministic packets significantly affects network performance because the fragmented deterministic packets not only introduce extra packet headers but also interrupt best-effort packets. Therefore, given a network and a flow set, our proposed joint algorithm of message fragmentation and scheduling is to minimize the number of packets such that the following two requirements can be satisfied.

- *Real-time requirement:* All messages must be delivered to destinations before their absolute deadlines.
- *No-conflict requirement:* A unidirectional link cannot serve more than one packet at the same time.

A flow set is called *schedulable* if it has a feasible schedule that meets all the requirements. A solution is called *optimal* if there are no other solutions with better objective values. When only linear networks are considered, and messages are too small to be fragmented, our problem is the same as the scheduling problem studied in [37]. Since the problem in [37] is NP-hard, our problem is also NP-hard. Therefore, to solve the problem, we propose two methods as shown in Fig. 4. First, the problem is formulated as an OMT specification [38] (in Section IV). Thus, if there exists a feasible solution, OMT solvers can find it. However, for some complex networks, the solvers cannot find any feasible solution in an acceptable time. Then, to improve the scalability of our algorithms, a heuristic algorithm is proposed (in Section V).

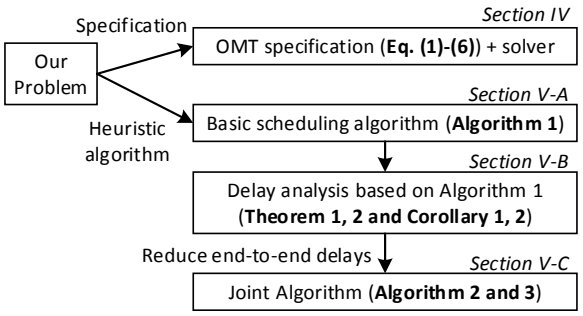


Fig. 4: Overview

IV. OMT Specification

Satisfiability modulo theories (SMT) have been widely used to determine whether a specification is satisfiable or not [39]. An OMT is an extension of a SMT and allows finding an optimal objective based on a SMT specification.

The inputs of the problem are network N and flow set F. To formulate the problem, we assume that a message is fragmented into, at most, U packets, which are denoted as $\{\tau_{i,j,1}, \tau_{i,j,2}, ..., \tau_{i,j,U}\}$. U is given by users. Three decision variables $w_{i,j,g}$, $s_{i,j,g}$ and $u_{i,j,g}$ are introduced. $w_{i,j,g}$ denotes the injection time of packet $\tau_{i,j,g}$. The size of packet $\tau_{i,j,g}$ is $s_{i,j,g}$ bytes. If $s_{i,j,g} = 0$, $\tau_{i,j,g}$ is defined as *invalid*; otherwise, it is valid. The objective of this work is to minimize the number of valid packets. To calculate the number of valid packets, we use $u_{i,j,g}$ to denote whether a packet is valid or not. If $\tau_{i,j,g}$ is valid, i.e., $s_{i,j,g} > 0$, then $u_{i,j,g} = 1$; otherwise, $u_{i,j,g} = 0$. Thus, the objective function is

$$\min \sum_{\forall f_i \in F, \forall j \in [0, \frac{H}{p_i}), \forall g \in [0,U)} u_{i,j,g}. \tag{1}$$

The minimizing problem respects the following constraints.

1) *Range constraint:* The ranges of the variables used in the proposed model are the following:

$$\begin{aligned} &\forall f_i \in F, \forall j \in [0, \frac{H}{p_i}), \forall g \in [0, U), \\ &0 \le s_{i,j,g} \le MSS, u_{i,j,g} \in \{0, 1\}, \\ &j \times p_i \le w_{i,j,g} \le j \times p_i + d_i, \\ &w_{i,j,g} \le w_{i,j,g+1}, u_{i,j,g} \ge u_{i,j,g+1}. \end{aligned} \tag{2}$$

If the size of a packet is greater than MSS, the packet has to be fragmented again on the IP layer. Thus, the upper bound of the packet size is MSS. The injection time of a packet must be between its generation time and deadline. To reduce the search space, we specify the order of packets: the packets are injected in increasing order of ID, and valid packets have smaller IDs than invalid packets.

2) *Size constraint:* A message is fragmented into several packets, and the sum of the size of these packets is equal to the size of the message. For each packet, based on the definition of $u_{i,j,g}$, if the size of the packet is 0, the corresponding $u_{i,j,g}$ is also 0; otherwise, it is 1. The size constraints are as follows:

$$\begin{aligned} \forall f_i \in F, \forall j \in [0, \frac{H}{p_i}), \sum_{\forall g \in [0,U)} s_{i,j,g} = \mathfrak{s}_i, \\ \forall g \in [0, U), ((s_{i,j,g} > 0) \wedge (u_{i,j,g} = 1)) \vee \\ ((s_{i,j,g} = 0) \wedge (u_{i,j,g} = 0)). \end{aligned} \tag{3}$$

3) *Real-time constraint:* If a packet is invalid, i.e., $\neg u_{i,j,g}$ is true, its real-time constraints do not need to be considered; otherwise, its arrival time cannot be later than its deadline. The arrival time of a packet is equal to the injection time plus the transmission time ($\frac{s_{i,j,g}+e}{v} \times r_i$), where e is the size of a packet header. Thus, the real-time constraints are as follows:

$$\begin{aligned} &\forall f_i \in F, \forall j \in [0, \frac{H}{p_i}), \forall g \in [0, U), \\ &\neg u_{i,j,g} \vee (w_{i,j,g} + \frac{s_{i,j,g} + e}{v} \times r_i \le j \times p_i + d_i). \end{aligned} \tag{4}$$

4) *No-conflict constraint:* For any two packets, under the following three conditions, we do not need to consider the conflict between them. First, the two packets are the same packet, i.e., for $\tau_{i,j,g}$ and $\tau_{x,y,z}$, $(i = x) \wedge (j = y) \wedge (g = z)$ is true. Second, not both packets are valid, i.e., $\neg u_{i,j,g} \vee \neg u_{x,y,z}$ is true. Third, the two packets do not pass through the same directed link, i.e., $(l_{a,b} \notin \Pi_i) \vee (l_{a,b} \notin \Pi_x)$ is true. If the two packets do not satisfy any of the above conditions, the time intervals when they occupy the same link cannot overlap;

otherwise, they conflict with each other. The constraints are as follows:

$$\begin{aligned}
&\forall l_{a,b} \in L, \forall f_i, f_x \in F, \forall j \in [0, \frac{H}{p_i}),\\
&\forall y \in [0, \frac{H}{p_x}), \forall g, z \in [0, U),\\
&((i = x) \wedge (j = y) \wedge (g = z)) \vee\\
&\neg u_{i,j,g} \vee \neg u_{x,y,z} \vee (l_{a,b} \notin \Pi_i) \vee (l_{a,b} \notin \Pi_x) \vee\\
&(A_{x,y,z}(l_{a,b}) + \frac{s_{x,y,z} + e}{v} < A_{i,j,g}(l_{a,b})) \vee\\
&(A_{i,j,g}(l_{a,b}) + \frac{s_{i,j,g} + e}{v} < A_{x,y,z}(l_{a,b})),
\end{aligned} \tag{5}$$

where $A_{i,j,g}(l_{a,b})$ denotes the time of starting to transmit the packet on link $l_{a,b}$, i.e.,

$$A_{i,j,g}(l_{a,b}) = w_{i,j,g} + \frac{s_{i,j,g} + e}{v} \times (q_{i,a,b} - 1). \tag{6}$$

$q_{i,a,b}$ denotes that $l_{a,b}$ is the $q_{i,a,b}$-th hop in path Π_i. Then, $A_{i,j,g}(l_{a,b}) + \frac{s_{i,j,g}+e}{v}$ is the finish time of the packet on link $l_{a,b}$.

OMT solvers can find the solution of the above specification. However, for complex problems, the execution time of off-the-shelf solvers is unacceptable. Therefore, in the next section, a fast algorithm is proposed to solve the problem.

V. Proposed Algorithm

Our algorithms schedule messages based on fixed priority scheduling [40], which is effective and has been widely used in real-time systems. Under fixed-priority scheduling, each message has an unique priority, and all the messages are scheduled in decreasing order of their priorities. For easy understanding, first, in Section V-A, we assume that priorities and message fragmentation have been determined. Based on the precondition, a basic fixed-priority scheduling algorithm (Algorithm 1) can calculate the injection times for all packets. Then, in Section V-B, based on the basic scheduling algorithm, the worst-case delays of messages are analyzed in order to determine what strategy can improve the real-time performance. Finally, in Section V-C, based on the analysis, our joint algorithm to fragment messages and schedule packets is proposed (as shown in Algorithm 3).

A. Basic Scheduling

To find the main factors that affect transmission delay, how packets are scheduled is presented in this section.

Algorithm 1 is used to calculate the injection time for each packet. Each message $m_{i,j}$ has a priority $\rho_{i,j}$. If $\rho_{i,j} < \rho_{x,y}$, then the scheduling algorithm calculates the injection time $w_{i,j,g}$ of the packets belonging to message $m_{i,j}$ first. Note that in Algorithm 1 it is assumed that all priorities $\rho_{i,j}$ and message fragmentation $\{\tau_{i,j,1}, \tau_{i,j,2}, ...\}$ have been obtained. The method of obtaining these is introduced in Section V-C.

Γ contains all of the unscheduled messages (lines 1 and 4). In each iteration (lines 2–10), Algorithm 1 calculates the injection time of the highest-priority message in Γ until there are no unscheduled messages. For message $m_{i,j}$, the set $M_{i,j}$ includes all of its valid packets. The packets in $M_{i,j}$ are assigned injection times (lines 5–8) separately. If at time t a packet can be scheduled without conflict (the function $NoConflict()$ in line 7), its injection time is t (line 8). The function $NoConflict(t, \tau_{i,j,g})$ returns true if, when $\tau_{i,j,g}$ is injected into the network at time t, there is no conflict between $\tau_{i,j,g}$ and the other scheduled packets.

Algorithm 1 Initial scheduling algorithm

Input: $< N, L >$, F, $\forall \rho_{i,j}$, $\forall s_{i,j,g}$, $\forall u_{i,j,g}$
Output: $\forall w_{i,j,g}$

```
1: ∀f_i ∈ F, ∀j ∈ [0, H/p_i), Γ = Γ + {m_{i,j}};
2: while Γ ≠ ∅ do
3:    m_{i,j} with the smallest ρ_{i,j} is the highest-priority message in Γ;
4:    Γ = Γ − {m_{i,j}};
5:    for each τ_{i,j,g} ∈ M_{i,j} do
6:       for t = (j × p_i) to (j × p_i + d_i − (s_{i,j,g}+e)/v × r_i) do
7:          if NoConflict(t, τ_{i,j,g}) then
8:             w_{i,j,g} = t; break;
9:       if τ_{i,j,g} is not scheduled then
10:         return FAIL;
11: return ∀w_{i,j,g};
```

Then, we analyze the time complexity of Algorithm 1. Lines 4 and 8–10 take constant time. The number of iterations of the **while** loop in line 2 and the **for** loop in lines 5 and 6 are, at most, $\sum_{\forall f_i \in F} \frac{H}{p_i}$, U, and p_i, respectively. The time complexity of $NoConflict()$ is $O(|N|)$. Line 1 is less complex than the other lines and is ignored. Therefore, the time complexity of Algorithm 1 is determined by lines 2, 5, 6 and $NoConflict()$, i.e., $O(\sum_{\forall f_i \in F} \frac{H}{p_i} \times U \times p_i \times |N|) = O(|F| \times H \times U \times |N|)$, where U is given by users.

B. Delay Analysis

The end-to-end delay of a message is defined as the time duration between its generation time and its received time. In the following, analyzing the end-to-end delay of the message $m_{i,j}$ is considered. Based on Algorithm 1, it is known that the messages in the conflict set $\Omega(m_{i,j})$ (Definition 1) determine the delay of $m_{i,j}$. When all the messages of $\Omega(m_{i,j})$ are transmitted in the active interval of $m_{i,j}$ and before the transmissions of $m_{i,j}$, the worst-case delay of $m_{i,j}$ occurs. $S(m_{i,j})$ is used to denote the packets that belong to the messages of $\Omega(m_{i,j})$, and $S(m_{i,j}) = \{\tau_{a,b,g} | \forall \tau_{a,b,g} \in M_{a,b}, \forall m_{a,b} \in \Omega(m_{i,j})\}$.

Definition 1. *$\Omega(m_{i,j})$ includes all of the messages $m_{a,b}$ that satisfy the following conditions:*

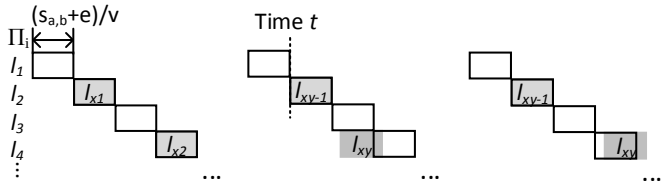
- *$m_{a,b}$ has a higher priority than $m_{i,j}$, i.e., $\rho_{a,b} < \rho_{i,j}$.*
- *$m_{a,b}$ and $m_{i,j}$ use the same links, i.e., $\Pi_a \cap \Pi_i \neq \emptyset$.*
- *the active intervals of $m_{a,b}$ and $m_{i,j}$ overlap each other, i.e., $[b \times p_a, b \times p_a + d_a) \cap [j \times p_i, j \times p_i + d_i) \neq \emptyset$.*

Based on Definition 1, it is known that the higher-priority messages in $\Omega(m_{i,j})$ pass through the links of Π_i, but maybe not all of them. However, as long as one of the hops of $m_{i,j}$ is delayed by the higher-priority messages, the entire $m_{i,j}$

is delayed. This is the same as transmitting higher-priority messages over the entire path. Hence, to bound the worst case, all of the higher-priority messages are extended to the entire path Π_i, i.e., when the delay of $m_{i,j}$ is considered, the routing path of a higher-priority message $m_{a,b}$ is also Π_i. This path extension does not result in an overestimation of the worst case or an underestimation of the conflicts from the higher-priority messages (as shown in Theorem 1). Intuitively, changing the path will cause more delay, or some conflicts may be neglected. However, in Theorem 1, we proved that the above situations do not exist.

Theorem 1. *In the proposed model, the path extension for the messages in set $\Omega(m_{i,j})$ does not exacerbate the worst-case delay and does not lose the conflicts from the higher-priority messages.*

Proof. First, the worst case is discussed. The path extension does not allow messages to be transmitted in parallel since all of them occupy the same links. Owing to the conflicts introduced by other messages, when the delay of $m_{i,j}$ is analyzed, the start times of the higher-priority messages in $\Omega(m_{i,j})$ are unknown and can be any time. Thus, even though the paths are not extended, it is possible that the transmission times of the messages in $\Omega(m_{i,j})$ do not overlap each other, i.e., they are serial. Therefore, the path extension does not exacerbate the worst case.



(a) Path extension (b) Assumption 1 (c) Assumption 2

Fig. 5: Illustration of higher-priority message $m_{a,b}$

Then, the conflicts from the higher-priority message $m_{a,b}$ are discussed. It is assumed that $\Pi_i = \{l_1, ..., l_{r_i}\}$, and $\Pi_i \cap \Pi_a = \{l_{x_1}, l_{x_2}, ..., l_{x_\varphi}\}$ $(1 \leq x_1 < x_2 < ... < x_\varphi \leq r_i)$. After extending the path, the transmissions of $m_{a,b}$ are shown in Fig. 5(a). To simplify the description, a message includes only one packet. If a message includes multiple packets, our proof is still available. It is assumed, to the contrary, that the extended path leads to reducing the conflicts, i.e., part of the original transmissions of $m_{a,b}$ that cannot be covered by the extended transmissions introduces more conflicts. When there exists only one link l_{x_1} in $\Pi_i \cap \Pi_a$, only one transmission must be covered by the extended transmission since the link is in the extended path and the message size is unchanged. When multiple links are in $\Pi_i \cap \Pi_a$, the transmission on l_{x_1} can be covered by the extended transmissions. It is assumed that the transmission on l_{x_y} $(1 < y \leq \varphi)$ is the first transmission that cannot be covered (as shown in Figs. 5(b) and (c)). Note that the proposed model adopts the shortest routing paths. The number of the hops between $l_{x_{y-1}}$ and l_{x_y} in Π_a is the same as that in Π_i, i.e., $x_y - x_{y-1} - 1$. Then, the start time of the transmission on l_{x_y} is $t + (x_y - x_{y-1} - 1) \times \frac{s_{a,b}+e}{v}$, where t is the end time of the transmission on $l_{x_{y-1}}$. This time is equal to the start time of the transmission on the extended path, which contradicts our assumption. Therefore, our path extension does not underestimate the conflicts from higher-priority messages. □

A packet set $\vec{S}(m_{i,j})$ is used to denote the transmitting order of packets. Set $\vec{S}(m_{i,j})$ includes two parts: the ordered packets of $S(m_{i,j})$ denoted by $\{\tau_1, \tau_2, ..., \tau_\sigma\}$, where $\sigma = |S(m_{i,j})|$, and the packets of $m_{i,j}$ denoted by $\{\tau_{\sigma+1}, ..., \tau_{\sigma+\varsigma}\}$, where $\varsigma = |M_{i,j}|$. For any two packets τ_k and τ_{k+1}, τ_k is transmitted before τ_{k+1}. Then, the worst case delay of $m_{i,j}$ can be calculated based on Theorem 2.

Theorem 2. *The generation time of $m_{i,j}$ is set to time 0, and the worst-case delay of $m_{i,j}$ is equal to the relative finish time of the last packet $\tau_{\sigma+\varsigma}$ bounded by a recursive function*

$$\begin{aligned} Delay(m_{i,j}) &= Finish(\tau_{\sigma+\varsigma}) \\ &= Finish(\tau_{\sigma+\varsigma-1}) + \delta(\tau_{\sigma+\varsigma}), \end{aligned} \tag{7}$$

where

$$Finish(\tau_1) = r_i \times \frac{s_1 + e}{v} \tag{8}$$

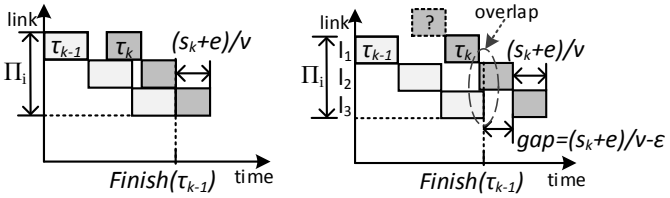
and

$$\delta(\tau_k) = \begin{cases} \frac{s_k+e}{v} + (\frac{s_k+e}{v} - \varepsilon) & \text{if } k \leq \sigma \text{ and } s_k \leq s_{k-1}, \\ r_i \times \frac{s_k+e}{v} - (r_i - 1) \times \frac{s_{k-1}+e}{v} + (\frac{s_{k-1}+e}{v} - \varepsilon) & \text{if } k \leq \sigma \text{ and } s_k > s_{k-1}, \\ \frac{s_k+e}{v} & \text{if } k > \sigma \text{ and } s_k \leq s_{k-1}, \\ r_i \times \frac{s_k+e}{v} - (r_i - 1) \times \frac{s_{k-1}+e}{v} & \text{if } k > \sigma \text{ and } s_k > s_{k-1}. \end{cases} \tag{9}$$

Proof. The finish time of a packet is equal to the finish time of its previous packet plus its own delay δ.

First, the calculation in the subset $\{\tau_1, ..., \tau_\sigma\}$ is explained. For the first packet τ_1, its finish time is equal to its transmission time (Eq. (8)). Then, for packet τ_k $(2 \leq k \leq \sigma)$, the following two conditions correspond to the first two lines of $\delta(\tau_k)$.

Condition 1: $k \leq \sigma$ and $s_k \leq s_{k-1}$. If there is no other conflict outside Π_i, τ_k should be transmitted after τ_{k-1} in succession (as shown in Fig. 6(a)). However, some external factors, such as the path outside Π_i and the unknown relationship between the generation time of τ_k and transmission time of τ_{k-1}, may make τ_k more delayed. This additional delay is called a *gap* (as shown in Fig. 6(b)). When the gap length is less than $\frac{s_k+e}{v}$, the time duration of τ_k occupying l_2 overlaps the time duration of τ_{k-1} occupying l_3. Thus, between τ_k and τ_{k-1}, no packet can be transmitted continuously on l_2 and l_3. The gap can only be idle. In the worst case, the gap length is $\frac{s_k+e}{v} - \varepsilon$, where ε approaches 0. If the gap length is greater $\frac{s_k+e}{v} - \varepsilon$, then l_2 and l_3 can be used continuously, and a subsequent packet can be inserted. Although it is possible that due to the delays introduced by external factors the subsequent packets in $\vec{S}(m_{i,j})$ cannot be inserted, the packets of $m_{i,j}$ can be inserted because they do not have the path outside Π_i. Therefore, the delay of $\delta(\tau_k)$ is its transmission delay plus the worst-case gap.

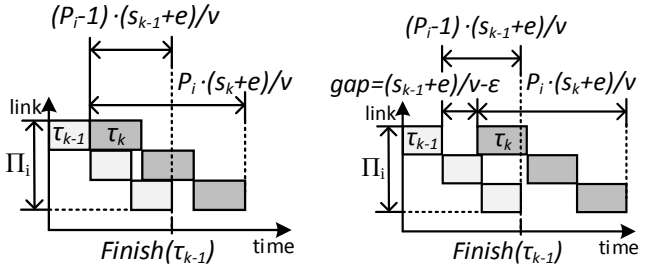


(a) Best case (b) Worst case

Fig. 6: Illustration of Condition 1

Condition 2: $k \leq \sigma$ and $s_k > s_{k-1}$. If the two packets can be transmitted in succession, the finish time of τ_k should be $Finish(\tau_{k-1}) + r_i \times \frac{s_k+e}{v} - (r_i - 1) \times \frac{s_{k-1}+e}{v}$ (as shown in Fig. 7(a)). However, due to unknown conflict, there may be a gap $\frac{s_{k-1}+e}{v} - \varepsilon$ between the two packets (as shown in Fig. 7(b)). Hence, the finish time of τ_k is

$$Finish(\tau_{k-1}) + r_i \times \frac{s_k + e}{v} - (r_i - 1) \times \frac{s_{k-1} + e}{v} + (\frac{s_{k-1} + e}{v} - \varepsilon).$$



(a) Best case (b) Worst case

Fig. 7: Illustration of Condition 2

Then, for the packets of $m_{i,j}$, since all of their conflicts have been included in $\vec{S}(m_{i,j})$, no unknown conflict results in gaps. Thus, in *Conditions 3 and 4*, δ_k does not contain gap delays, and the others are the same as *Conditions 1 and 2*. □

In Theorem 2, the impact of fragmentation and scheduling on the worst-case delay is formulated. Based on the theorem, we can prove Corollary 1 and 2. Then, the worst-case delay can be reduced by constructing a proper solution based on the two corollaries.

Corollary 1. *When the packets of $\vec{S}(m_{i,j})$ are in a monotonic non-increasing order of sizes, the minimum value of $Delay(m_{i,j})$ can be obtained as follows:*

$$Delay(m_{i,j}) = (r_i - 2) \times \frac{s_{max} + e}{v} + 2 \times \sum_{\forall \tau_k \in S(m_{i,j})} \frac{s_k + e}{v} + \sum_{\forall \tau_k \in M_{i,j}} \frac{s_k + e}{v} - \varepsilon, \quad (10)$$

where s_{max} is the maximum size in $\vec{S}(m_{i,j})$.

Proof. First, we discuss $S(m_{i,j})$ and $M_{i,j}$, respectively, and then consider the relationship between them.

For $S(m_{i,j})$, the first two conditions of Eq. (9) are rewritten as follows.

$$\delta(\tau_k) = 2\frac{s_k + e}{v} - \varepsilon + \begin{cases} 0 \text{ if } k \leq \sigma \text{ and } s_k \leq s_{k-1}, \\ (r_i - 2) \times (\frac{s_k+e}{v} - \frac{s_{k-1}+e}{v}) \\ \text{ if } k \leq \sigma \text{ and } s_k > s_{k-1}. \end{cases} \quad (11)$$

Condition 2 is not less than zero because $r_i \geq 2$ and $s_k > s_{k-1}$. Therefore, when all packets satisfy Condition 1, i.e., the packets are transmitted in a monotonic non-increasing order of sizes, the value of $Finish(\tau_\sigma)$ is the minimum as follows

$$Finish(\tau_\sigma) = Finish(\tau_1) + \sum_{\forall \tau_k \in S(m_{i,j}) - \{s_1\}} 2\frac{s_k + e}{v} - \varepsilon. \quad (12)$$

For $M_{i,j}$, when all of its packets are transmitted without gaps, the minimum delay occurs, i.e., $\sum_{\forall \tau_k \in M_{i,j}} \frac{s_k+e}{v}$. This is the same as Condition 3 of Eq. (9) in which the packets are in a monotonic non-increasing order of sizes.

Then, we consider the relationship between τ_σ (the last packet of $S(m_{i,j})$) and $\tau_{\sigma+1}$ (the first packet of $M_{i,j}$).

If $s_{\sigma+1} \leq s_\sigma$, then

$$\begin{aligned} Delay_{s_{\sigma+1} \leq s_\sigma}(m_{i,j}) &= Finish(\tau_\sigma) + \sum_{\forall \tau_k \in M_{i,j}} \frac{s_k + e}{v} \\ &= (r_i - 2) \times \frac{s_1 + e}{v} + 2 \times \sum_{\forall \tau_k \in S(m_{i,j})} \frac{s_k + e}{v} \\ &+ \sum_{\forall \tau_k \in M_{i,j}} \frac{s_k + e}{v} - \varepsilon. \end{aligned} \quad (13)$$

If $s_{\sigma+1} > s_\sigma$, then

$$\begin{aligned} Delay_{s_{\sigma+1} > s_\sigma}(m_{i,j}) &= Finish(\tau_\sigma) + r_i \times \frac{s_{\sigma+1} + e}{v} \\ &- (r_i - 1) \times \frac{s_\sigma + e}{v} + \sum_{\forall \tau_k \in M_{i,j} - \{\tau_{\sigma+1}\}} \frac{s_k + e}{v} \\ &= Finish(\tau_\sigma) + \sum_{\forall \tau_k \in M_{i,j}} \frac{s_k + e}{v} \\ &+ (r_i - 1) \times \frac{s_{\sigma+1} + e}{v} - (r_i - 1) \times \frac{s_\sigma + e}{v} \\ &= Delay_{s_{\sigma+1} \leq s_\sigma}(m_{i,j}) + (r_i - 1) \times \frac{s_{\sigma+1} + e}{v} \\ &- (r_i - 1) \times \frac{s_\sigma + e}{v}. \end{aligned} \quad (14)$$

Hence, $Delay_{s_{\sigma+1} \leq s_\sigma}(m_{i,j}) < Delay_{s_{\sigma+1} > s_\sigma}(m_{i,j})$ since $(r_i - 1) \times \frac{s_{\sigma+1}+e}{v} > (r_i - 1) \times \frac{s_\sigma+e}{v}$. Therefore, when all the packets are in a monotonic non-increasing order of sizes, Eq. (13) can be used to calculate the minimum delay, where s_1 denotes the maximum size. □

Corollary 2. *Given a fixed number of packets, the value of $Delay(m_{i,j})$ can be further decreased by minimizing s_{max}.*

Proof. For the right-hand side of Eq. (10), the fourth term approaches zero and can be ignored. The second term is rewritten as

$$\frac{2}{v} \times \sum_{\forall m_{a,b} \in \Omega(m_{i,j})} (s_{a,b} + |M_{a,b}| \times e).$$

Since $|M_{a,b}|$ is given, the second term is a fixed value. Similarly, the third term is also a fixed value. Therefore, only s_{max} in the first term can be reduced to decrease the value of $Delay(m_{i,j})$. □

C. Joint Algorithm

The proposed joint algorithm is based on Corollary 1 and 2. Packets are scheduled in non-increasing order of their sizes as described in Corollary 1. Thus, only when messages cannot be scheduled, the size of packets is reduced to improve parallelism, and the number of packets is increased. This also conforms to our objective of minimizing the number of packets (Eq. (1)). Then, messages are fragmented into packets of $\bar{s}$ bytes, where $\bar{s}$ is a variable and corresponds to s_{max} of Corollary 1. If message $m_{i,j}$ is unschedulable, the schedules of $S(m_{i,j})$ are rolled back, and $\bar{s}$ is reduced such that the delay can be decreased as described in Corollary 2. In our heuristic algorithms, we do not need to consider invalid packets, which are only used in the OMT specification.

If the last packet of a message is less than $\bar{s}$ bytes, it is enlarged to $\bar{s}$ bytes to satisfy Corollary 1. Intuitively, enlarging packets will increase delay. However, if the packet is not enlarged, it may introduce more delay to the subsequent packets. For example, s' bytes are padded into a packet of s_k bytes. Then, based on Eq. (9), the delay of packet s_{k+1} is

$$\begin{aligned} D &= Finish(\tau_{k-1}) + 2 \times \frac{s_k + s' + e}{v} - \varepsilon + \frac{s_{k+1} + e}{v} \\ &= B + 2 \times \frac{s'}{v}, \end{aligned} \tag{15}$$

where

$$B = Finish(\tau_{k-1}) + 2 \times \frac{s_k + e}{v} - \varepsilon + \frac{s_{k+1} + e}{v}. \tag{16}$$

If the packet is not enlarged, the delay is

$$\begin{aligned} D' =& Finish(\tau_{k-1}) + 2 \times \frac{s_k + e}{v} - \varepsilon \\ &+ r_i \times \frac{s_{k+1} + e}{v} - (r_i - 1) \times \frac{s_k + e}{v} \\ =& B + (r_i - 1) \times \frac{s_{k+1} - s_k}{v}. \end{aligned} \tag{17}$$

The upper bounds of s' and $s_{k+1} - s_k$ are both $\bar{s}$. Then, $D < B + 2 \times \frac{\bar{s}}{v}$, and $D' < B + (r_i - 1) \times \frac{\bar{s}}{v}$. The upper bound of D is fixed, while D' increases with path length r_i. Therefore, when a path includes more than three hops, enlarging the packet to $\bar{s}$ can reduce the delay of the subsequent packets. As shown in Fig. 8, when τ_k is enlarged, the delay of τ_{k+1} is reduced. In Section VI, our evaluations also demonstrate this property (shown as ME+EN and JA-EN in Figs. 11-15).

The proposed joint algorithm is shown in Algorithm 3. First, function $PriorityAssignment()$ (Algorithm 2) is invoked to assign priorities. $PriorityAssignment()$ is an extension of the classical priority assignment in [41]. The classical priority assignment traverses priorities from lowest to highest (line 2) and assigns a priority to a task if the task satisfies the following condition: when all other unassigned tasks have higher priorities than the task, the delay of the task is less than its deadline (lines 3–6). Under this assignment policy,

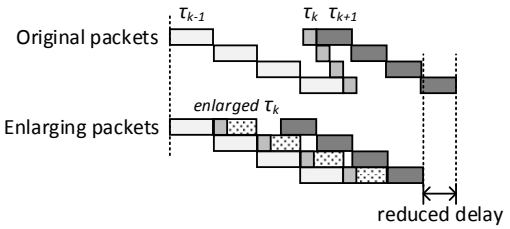


Fig. 8: Enlarging τ_k to reduce the delay of τ_{k+1}

Algorithm 2 $PriorityAssignment(F)$

Input: F
Output: $\vec{\Omega}$

1: $\forall f_i \in F, \forall j \in [0, \frac{H}{p_i}), \Gamma = \Gamma + \{m_{i,j}\}$;
2: **for** each $\rho = \sum_{\forall f_i \in F} \frac{H}{p_i}$ to 1 **do**
3: **for** each $m_{i,j} \in \Gamma$ **do**
4: assume that except $m_{i,j}$, all of the others in Γ have higher priorities than $m_{i,j}$;
5: **if** $Delay'(m_{i,j}) \leq j \times p_i + d_i$ **then**
6: $\rho_{i,j} = \rho$; $\Gamma = \Gamma - \{m_{i,j}\}$; break;
7: **else**
8: $\chi_{i,j} = Delay'(m_{i,j}) - (j \times p_i + d_i)$;
9: **if** no message is assigned ρ **then**
10: assign ρ to the message with the minimum $\chi_{i,j}$;
11: $\Gamma = \Gamma - \{m_{i,j}\}$;
12: store messages in decreasing order of priority in set $\vec{\Omega}$;
13: **return** $\vec{\Omega}$;

the initial order of all messages does not affect schedulability, which has been proved in [42], [41]. The classical priority assignment can reach an optimal solution only if the delay is exact. Our analyzed delay is the upper bound of the actual delay. Therefore, if the analyzed message delay is less than the deadline, then the assignment policy must make the message schedulable.

The prerequisite of calculating $Delay()$ is the known message fragmentation. However, before $PriorityAssignment()$, message fragmentation is not determined. Thus, $Delay()$ is changed as follows:

$$\begin{aligned} Delay'(m_{i,j}) = (r_i - 2) \times & \frac{MSS + e}{v} \\ + \frac{2}{v} \times & \sum_{\forall m_{a,b} \in \Omega(m_{i,j})} (s_{a,b} + \lceil \frac{s_{a,b}}{\lambda} \rceil \times e) \\ + \frac{1}{v} \times & (s_{i,j} + \lceil \frac{s_{i,j}}{\lambda} \rceil \times e) - \varepsilon, \end{aligned} \tag{18}$$

where λ is the minimum packet size and given by users. $Delay'()$ is the upper bound of $Delay()$ because MSS and $\lceil \frac{s_{a,b}}{\lambda} \rceil$ is the upper bound of s_{max} and $|M_{a,b}|$, respectively. Thus, if $Delay'()$ is less than the deadline, the current priority ρ can make message $m_{i,j}$ schedulable (lines 4–6); otherwise, the deviation, denoted as $\chi_{i,j}$, between delay and deadline is calculated (lines 7–8). Then, if the current priority is not assigned to any message, it is assigned to the message with the minimum $\chi_{i,j}$ (lines 7–11) because the message is most likely schedulable at the current priority level. Algorithm 2 returns

a message set $\vec{\Omega}$ that includes all of the messages sorted in decreasing order of priorities (lines 12–13).

Lines 4, 6, 9, 11 and 13 take constant time. The number of iterations of the **for** loop in lines 2 and 3 are $\sum_{\forall f_i \in F} \frac{H}{p_i}$. In $Delay'()$ and $Delay()$, the running time of the summations are $\sum_{\forall f_i \in F} \frac{H}{p_i}$ and $\sum_{\forall f_i \in F} \frac{H}{p_i} \times U$, respectively. In addition, the time complexities of lines 1, 10 and 12 are $O(\sum_{\forall f_i \in F} \frac{H}{p_i})$, $O(|F|)$, and $O(n \log n)$, respectively. However, they are less complex than the other lines and are ignored. Therefore, the time complexity of this function is determined by lines 2, 3 and $Delay()$, i.e., $O(M^3 \times U)$, where M is the total number of messages, and $M = \sum_{\forall f_i \in F} \frac{H}{p_i}$.

Algorithm 3 Joint algorithm

Input: F, Δ
Output: $\forall M_{i,j}$, $\forall w_{i,j,g}$

1: $\vec{\Omega} = PriorityAssignment(F)$;
2: $\bar{s}$ = MSS; $k = 1$;
3: **while** $k \leq |\vec{\Omega}|$ **do**
4: fragment the k-th message, denoted $m_{i,j}$, into multiple packets of $\bar{s}$ bytes;
5: **for** each $\tau_{i,j,g} \in M_{i,j}$ **do**
6: **for** $t = (j \times p_i)$ to $(j \times p_i + d_i - \frac{s_{i,j,g}+e}{v} \times r_i)$ **do**
7: **if** $NoConflict(t, \tau_{i,j,g})$ **then**
8: $w_{i,j,g} = t$; break;
9: **if** $\exists \tau_{i,j,g} \in M_{i,j}$ is not scheduled **then**
10: k = the smallest ID in $\Omega(m_{i,j})$;
11: $\forall g \in [k, |\vec{\Omega}|]$, clear the injection times of the gth message; *//schedules roll back*
12: $\bar{s} -= \Delta$;
13: **if** $\bar{s} < \lambda$ **then**
14: **return** FAIL;
15: **else**
16: $k++$;
17: **return** $\forall M_{i,j}$, $\forall w_{i,j,g}$;

Then, Algorithm 3 fragments and schedules messages in the order they are in set $\vec{\Omega}$ (lines 3–16). Messages are fragmented into packets of $\bar{s}$ bytes (line 4), and the range of $\bar{s}$ is from MSS to λ (lines 2 and 12–14). If a packet cannot be scheduled under the current $\bar{s}$ (line 9), $\bar{s}$ is reduced by a given step value Δ (line 12), and the messages that directly effect the packet are re-fragmented and re-scheduled (lines 10–11). The process of generating schedules (lines 5–8) is the same as that in Algorithm 1.

Owing to the roll-back operation (line 11), the time complexity of Algorithm 3 is pseudo-polynomial. The number of rollbacks is up to $\frac{MSS}{\Delta}$, and the time complexity of lines 3–7 has been calculated in Algorithm 1. Therefore, the time complexity of Algorithm 3 is $O(|F| \times H \times U \times |N| \times \frac{MSS}{\Delta})$, where U and Δ are given by users.

VI. Evaluation

In this section, we evaluate the proposed algorithms based on extensive test cases. Three metrics are used in our evaluation: (1) *schedulable ratio* is the percentage of test cases for which an algorithm can find a feasible solution; (2) *the number of packets* is our objective as shown in Eq. (1); and (3) *execution time* is the time required to generate a feasible solution.

TABLE I: parameters

n	Number of network nodes
f	Number of flows
p	Period range
s	Message size range
Δ	Step value to reduce packet size

The proposed joint algorithm (JA) and OMT-based method are compared with the following five methods: ME, ME+AD, ME+EN, JA-EN and BL.

- ME adopts the traditional Ethernet fragmentation and the earliest deadline first (EDF) scheduling algorithm [40].
- ME+AD is an extension of ME. If ME cannot schedule a test case under the initial MSS, ME+AD decreases the MSS by the same parameter Δ as JA and re-schedules the test case. Repeat this process until the test case can be scheduled.
- ME+EN is also an extension of ME. The only difference between ME+EN and ME is that ME+EN enlarges the last packets of messages, while ME does not.
- JA-EN is the same as JA except that it does not enlarge the last packets.
- BL is a baseline for comparisons. To illustrate the efficiency of JA, an optimal result is needed. However, in an acceptable time, off-the-shelf solvers can only find optimal results for simple problems. Thus, when optimal results cannot be found, an approximately optimal result is needed, which is BL in our evaluations. When a schedulable ratio is considered, BL is defined as the percentage of test cases that satisfy the following necessary condition: if all port utilizations are not larger than 100%, the test case is schedulable. When the comparison metric is the number of packets, BL is the sum of packets that are fragmented by the traditional Ethernet fragmentation. Thus, BL is better than or equal to the optimal result. If the proposed algorithm is close to BL, it must be close to the optimal results.

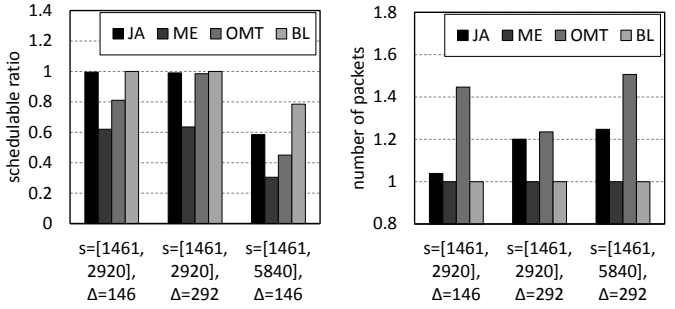
All algorithms are written in C and run on a Windows machine with a 2.5-GHz CPU and 8 GB of memory.

All test cases are generated based on the parameters shown in Table I. Each test case includes $\frac{n}{2}$ switches and $\frac{n}{2}$ end systems. Each switch has 4 ports, one of which is connected to an end system, and the remaining ports can be connected to other switches. The switch-end system pairs are randomly located on a two-dimensional coordinate system. Then, the switches are traversed in increasing order of ID. For each switch, the free ports are connected to the nearest switches that still have free ports. To illustrate the universality of our algorithms, we do not limit the network topology. There are f flows. The flows randomly select their source and destination end systems. Routing paths are generated using the Dijkstra algorithm. To restrict the length of the hyperperiod, we adopt harmonic periods. Each period is a random number that is in the period range p and conforms to the expression $400\mu s \times 2^n$,

where n is a non-negative integer. Message sizes are random numbers in the message size range s. The deadline of a flow is also a random number in the range $[\frac{p_i}{2}, p_i]$. The transmission speed $v = 31\ bytes/\mu s$ is measured in our simple prototype network such that the logical time of test cases can be mapped to the physical time. In the following comparisons, these parameters are changed and extensive test cases are generated to evaluate the efficiency and universality of the proposed algorithm. In Section VI-A, we compare OMT with JA, ME and BL. Then, due to the long execution time of the OMT solver, in Section VI-B, except OMT, the other algorithms are fully evaluated.

A. Evaluations with OMT

The Microsoft solver Z3 [43] is invoked to solve the OMT specification. The time limit of Z3 is set as 600s. To make test cases solvable, $n = 4$, $f = 4$, and $p = \{400\mu s, 800\mu s\}$ are set. The other parameters are shown in Fig. 9. For each parameter setting, 200 test cases are generated. Fig. 9 (a) and (b) show schedulable ratios and the number of packets normalized with BL, respectively. Since ME adopts traditional fragmentation, it does not introduce extra packets. However, the traditional algorithm has the worst schedulable ratio. OMT is not better than JA because Z3 cannot find optimal results for some test cases within the time limit. The schedulable ratio of JA is close to BL. This illustrates that JA can significantly improve the schedulability. Other than OMT, the other algorithms can finish execution within 10 ms. The execution time of OMT is shown in Fig. 10. Most of the test cases can be solved by Z3 in 300 s. Once the execution time exceeds 300 s, it is difficult to solve these test cases in 600 s. The test cases in Fig. 10 (b) have short execution times because they include the minimum number of packets, i.e., the minimum number of variables. Therefore, if a problem includes a small number of variables, OMT can be adopted to find optimal results; otherwise, JA is the best choice.

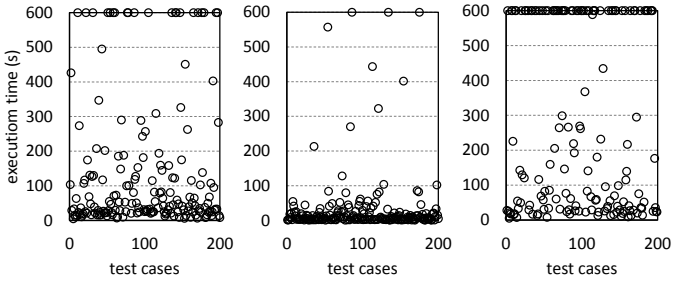


(a) Schedulable ratio (b) Number of packets

Fig. 9: Evaluations of OMT

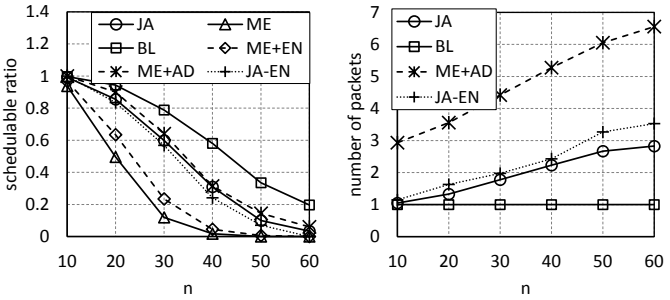
B. Evaluations without OMT

To fully evaluate the proposed algorithm, in the following, each parameter is changed over a larger range to generate test cases. The basic setting is shown in the caption of Fig. 11. Then, Figs. 12, 13, 14, and 15 change the period parameter, size parameter, step value, and number of flows, respectively, to illustrate the efficiency and universality of JA. For each parameter setting, 2000 test cases are generated.



(a) $1461 \leq s \leq 2920, \Delta = 146$ (b) $1461 \leq s \leq 2920, \Delta = 292$ (c) $1461 \leq s \leq 5840, \Delta = 146$

Fig. 10: Execution time of OMT

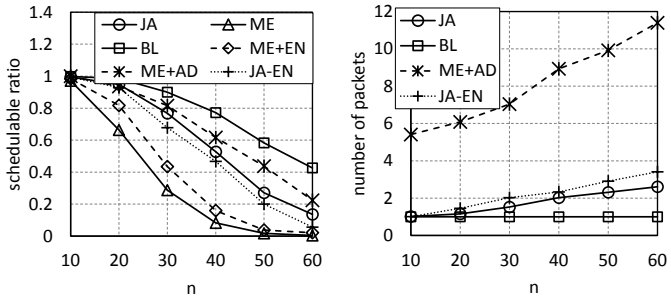


(a) Schedulable ratio (b) Number of packets

Fig. 11: Comparisons under varying n, $p = [800\mu s, 6.4ms]$, $s = [1461, 5480]$, $\Delta = 146$, and $f = n$

Fig. 11 shows the comparison with a varying the number of network nodes. The schedulable ratio decreases as the number of nodes increases. This is because the greater the number of nodes, the more difficult the scheduling becomes. ME+EN is better than ME. This is because ME makes more resource fragmentations unusable and cannot provide a long enough time duration to transmit no-wait packets. Fig. 11 (b) shows the number of packets normalized with BL. ME and ME+EN are not shown since they both adopt the traditional fragmentation and have the same number of packets with BL. The larger the network, the longer the routing path. Thus, when $n = 60$, to make flows schedulable, smaller packets are needed to improve the parallelism such that the delay introduced by the long path can be reduced. Therefore, the number of packets increases as n increases. Compared to ME, although JA introduces more packets, it still can find feasible solutions. This is because, in ME, different packet sizes cause excessive resource fragmentations. ME+AD uses many small packets to improve the parallelism. Thus, the schedulable ratio of ME+AD is slightly better than JA, but the number of packets of ME+AD is more than double that of JA. Therefore, in a network, if there are sufficient resources, the smaller MSS should be adopted to improve the real-time performance. Compared to JA, JA-EN causes more resource fragmentations. Hence, even though part of messages are fragmented into

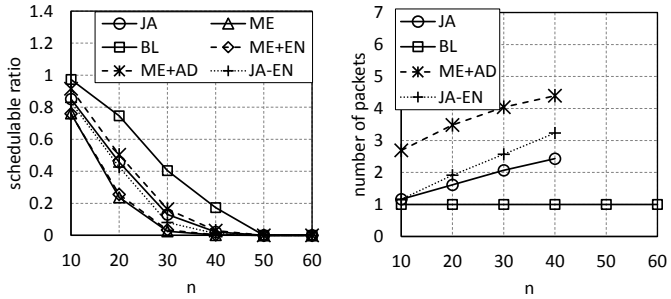
smaller packets to improve the parallelism, the schedulable ratio of JA-EN is still less than that of JA.



(a) Schedulable ratio (b) Number of packets

Fig. 12: Comparisons under varying n, $p = [800\mu s, \underline{12.8ms}]$, $s = [1461, 5480]$, $\Delta = 146$, and $f = n$

The period range is extended to illustrate its effect on schedulable ratios and the number of packets, as shown in Fig. 12. Compared to Fig. 11(a), Fig. 12(a) shows higher schedulable ratios because more time resources can be utilized to make flows satisfy real-time constraints. Then, since the real-time constraints are easily satisfied, messages do not need to be fragmented into small packets to reduce the delay. Thus, in Fig. 12(b), the number of packets of JA is less than that in Fig. 11(b). ME+AD makes full use of the time resources to improve the schedulability but results in too many packets. Compared to ME+AD, JA not only guarantees the requirements of deterministic communications but also reserves more resources for best-effort communications.

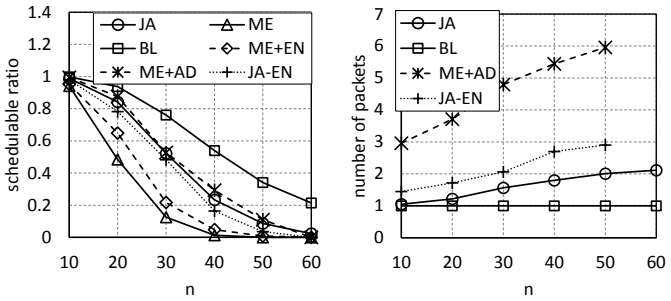


(a) Schedulable ratio (b) Number of packets

Fig. 13: Comparisons under varying n, $p = [800\mu s, 6.4ms]$, $s = [1461, \underline{8760}]$, $\Delta = 146$, and $f = n$

In Fig. 13, the message size is larger than that of other figures. Since large messages are difficult to schedule, the schedulable ratios in Fig. 13(a) are low. Compared to JA, although ME+AD improves the schedulable ratio by 3%, it consumes more than twice the resources. JA-EN is close to JA because large messages can be fragmented into more packets so that JA-EN searchs fesabile solutions in a larger solution space. When $n \geq 50$, no algorithm can schedule test cases, not even BL. Therefore, in real applications, the periods and deadlines of large messages must be extended to make them schedulable, as illustrated in Fig. 12.

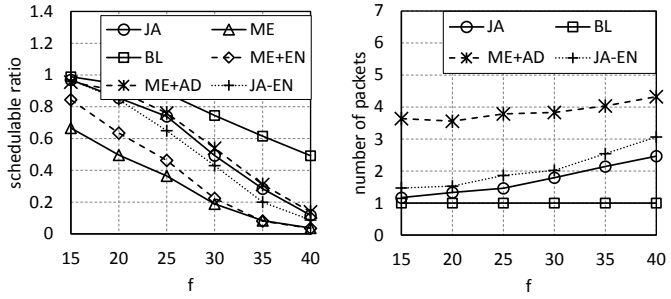
In Fig. 14, Δ is increased. Messages cannot be fragmented into small pieces. Both the schedulable ratio and the number



(a) Schedulable ratio (b) Number of packets

Fig. 14: Comparisons under varying n, $p = [800\mu s, 6.4ms]$, $s = [1461, 5480]$, $\Delta = \underline{438}$, and $f = n$

of packets are reduced. Other than Δ, if the other parameters are changed, test cases are changed accordingly. Only Δ is independent of test cases. Therefore, when given a real network, we can adjust Δ to trade off between the schedulable ratio and the number of packets.



(a) Schedulable ratio (b) Number of packets

Fig. 15: Comparison under varying f, $p = [800\mu s, 6.4ms]$, $s = [1461, 5480]$, $\Delta = 146$, and $n = 20$

Fig. 15 shows the comparison under varying f. As f increases, the increased packets make more test cases unschedulable, and thus the schedulable ratio decreases. The number of packets increases slowly, because the test cases that contain more packets are unschedulable, and these unschedulable test cases are not included in the figure. When there are not many flows, the schedulable ratio of ME+EN is significantly higher than that of ME. Therefore, ME+EN is an efficient method for low-load test cases.

VII. Conclusions

In this paper, to improve the real-time performance of networked control systems, the joint problem of message fragmentation and no-wait scheduling is explored. First, an OMT specification is proposed and then an off-the-shelf solver utilized to find its optimal solution. However, the OMT-based algorithm is limited by the execution time of the off-the-shelf solver. Then, to improve the scalability of our algorithm, a heuristic algorithm is proposed. The algorithm constructs low-delay fragmentation and schedules based on the message delay analysis. Finally, extensive test cases are conducted to evaluate the proposed algorithms. The results indicate that the proposed

joint algorithm outperforms existing ones. In the future, we will implement a TSN switch based on FPGA and evaluate our algorithms in a real TSN.

ACKNOWLEDGMENT

This work was partially supported by National Key Research and Development Program of China (2018YFB1700200), National Natural Science Foundation of China (61972389, 61903356, 61803368 and U1908212), Youth Innovation Promotion Association of the Chinese Academy of Sciences, National Science and Technology Major Project (2017ZX02101007-004), Liaoning Provincial Natural Science Foundation of China (2019-YQ-09, 20180520029 and 20180540114), and China Postdoctoral Science Foundation (2019M661156).

APPENDIX A
SYMBOL

N	Node set
L	Link set
n_i	The i-th node
$l_{i,j}$	The link between n_i and n_j
F	Flow set
f_i	The i-th flow
p_i	Period of f_i
d_i	Relative deadline of f_i
$\mathfrak{s}_i$	Message size of f_i
Π_i	Path of f_i
$m_{i,j}$	The j-th message of f_i
H	Hyperperiod
v	Transmission speed
r_i	Number of hops in path Π_i
U	A message is fragmented into, at most, U packets.
$\tau_{i,j,g}$	The g-th packet of $m_{i,j}$
$w_{i,j,g}$	Injection time of $\tau_{i,j,g}$
$s_{i,j,g}$	Size of $\tau_{i,j,g}$
$u_{i,j,g}$	If $u_{i,j,g} = 1$, $\tau_{i,j,g}$ is valid.
e	Size of a packet header
$A_{i,j,g}(l_{a,b})$	Time of starting to transmit $\tau_{i,j,g}$ on $l_{a,b}$
$q_{i,a,b}$	Location of $l_{a,b}$ in Π_i
$\rho_{i,j}$	Priority of $m_{i,j}$
Γ	Unscheduled message set
$M_{i,j}$	Valid packet set of $m_{i,j}$
t	Time variable
$\Omega(m_{i,j})$	Conflict set (Definition 1)
$S(m_{i,j})$	Set of packets that belong to $\Omega(m_{i,j})$
$\{l_1, ..., l_{r_i}\}$	Links in Π_i
φ	Number of links in $\Pi_i \cap \Pi_a$
l_{x_y}	The x_y-th link in $\Pi_i \cap \Pi_a$, and $y \in (1, \varphi]$
$\vec{S}(m_{i,j})$	Set of ordered packet
σ	Number of packets in $S(m_{i,j})$
ς	Number of packet in $M_{i,j}$
$\{\tau_1, ..., \tau_{\sigma+\varsigma}\}$	Packets in $\vec{S}(m_{i,j})$
ε	A value close to 0
s_{max}	The maximum size in $S(m_{i,j})$
$\bar{s}$, s'	Size variables
λ	The minimum packet size
$\chi_{i,j}$	The deviation between delay and deadline
$\vec{\Omega}$	Set of sorted messages
Δ	A step value used to adjust packet sizes

REFERENCES

[1] P. Pop, M. L. Raagaard, M. Gutierrez, and W. Steiner, "Enabling fog computing for industrial automation through time-sensitive networking (TSN)," *IEEE Commun. Stand. Mag.*, vol. 2, no. 2, pp. 55–61, Jul. 2018.

[2] J. Lee and S. Park, "Time-sensitive network (TSN) experiment in sensor-based integrated environment for autonomous driving," *Sensors*, vol. 19, no. 5, pp. 1–11, May 2019.

[3] L. Huang, Y. Liang, Y. Zhang, Y. Wang, and Q. Wang, "Time-sensitive network technology and its application in energy internet," in *Proc. 2019 IEEE Int. Conf. Energy Internet*, 2019, pp. 211–216.

[4] J. Jiang, Y. Li, S. H. Hong, A. Xu, and K. Wang, "A time-sensitive networking (TSN) simulation model based on OMNET++," in *Proc. IEEE Int. Conf. Mechatronics and Automation*, 2018, pp. 643–648.

[5] 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 Int. Conf. Real-Time Networks and Systems*, 2016, pp. 183–192.

[6] Cisco Systems, Inc. (2020) Cisco industrial ethernet 4000, 4010 and 5000 switch software configuration guide. [Online]. Available: https://www.cisco.com

[7] N. Semiconductors. (2016) SJA1105 product data sheet. [Online]. Available: https://www.nxp.com/docs/en/data-sheet/SJA1105.pdf

[8] F. Dürr and N. G. Nayak, "No-wait packet scheduling for IEEE time-sensitive networks (TSN)," in *Proc. 24th Int. Conf. Real-Time Networks and Systems*, 2016, pp. 203–212.

[9] X. Jin, F. Kong, L. Kong, W. Liu, and P. Zeng, "Reliability and temporality optimization for multiple coexisting WirelessHART networks in industrial environments," *IEEE Trans. Ind. Elect.*, vol. 64, no. 8, pp. 6591–6602, 2017.

[10] J. Tang, B. Shim, and T. Q. Quek, "Service multiplexing and revenue maximization in sliced C-RAN incorporated with URLLC and multicast eMBB," *IEEE J. Sel. Area. Commun.*, vol. 37, no. 4, pp. 881–895, 2019.

[11] L. Kong, M. Xia, X.-Y. Liu, G. Chen, Y. Gu, M.-Y. Wu, and X. Liu, "Data loss and reconstruction in wireless sensor networks," *IEEE Trans. Parall. and Distrib. Sys.*, vol. 25, no. 11, pp. 2818–2828, 2013.

[12] R. S. Oliver, S. S. Craciunas, and W. Steiner, "IEEE 802.1 Qbv gate control list synthesis using array theory encoding," in *Proc. IEEE Real-Time and Embedded Technology and Applications Symp.*, 2018, pp. 13–24.

[13] X. Jin, C. Xia, N. Guan, C. Xu, D. Li, Y. Yin, and P. Zeng, "Real-time scheduling of massive data in time sensitive networks with a limited number of schedule entries," *IEEE Access*, vol. 8, no. 1, pp. 6751–6767, 2020.

[14] N. G. Nayak, F. Dürr, and K. Rothermel, "Incremental flow scheduling and routing in time-sensitive software-defined networks," *IEEE Trans. Ind. Informat.*, vol. 14, no. 5, pp. 2066–2075, 5 2017.

[15] M. L. Raagaard, P. Pop, M. Gutiérrez, and W. Steiner, "Runtime reconfiguration of time-sensitive networking (TSN) schedules for fog computing," in *Proc. 2017 IEEE Fog World Congress*, 2017, pp. 1–6.

[16] V. Gavriluţ, L. Zhao, M. L. Raagaard, and P. Pop, "AVB-aware routing and scheduling of time-triggered traffic for TSN," *IEEE Access*, vol. 6, no. 11, pp. 75 229–75 243, 2018.

[17] J. Falk, F. Dürr, and K. Rothermel, "Exploring practical limitations of joint routing and scheduling for TSN with ILP," in *Proc. 24th International Conference on Embedded and Real-Time Computing Systems and Applications*, 2018, pp. 136–146.

[18] 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, no. 4, pp. 44 165–44 181, 2019.

[19] L. Zhao, P. Pop, Z. Zheng, and Q. Li, "Timing analysis of AVB traffic in TSN networks using network calculus," in *Proc. 2018 IEEE Real-Time and Embedded Technology and Applications Symp.*, 2018, pp. 25–36.

[20] I. Ferretti and L. Zavanella, "Batch energy scheduling problem with no-wait/blocking constraints for the general flow-shop problem," *Procedia Manufacturing*, vol. 42, no. 4, pp. 273–280, 2020.

[21] F. Della Croce, A. Grosso, and F. Salassa, "Minimizing total completion time in the two-machine no-idle no-wait flow shop problem," *J. Heuristics*, vol. 19, no. 10, pp. 1–15, 2019.

[22] X. Wang, K. Xing, Y. Feng, and Y. Wu, "Scheduling of flexible manufacturing systems subject to no-wait constraints via Petri nets and heuristic search," *IEEE Trans. Syst., Man, and Cybern.: Syst.*, vol. 49, no. 12, pp. 1–12, 2019.

[23] M. Behnam, R. Marau, and P. Pedreiras, "Analysis and optimization of the MTU in real-time communications over switched ethernet," in *Proc. 16th Conf. Emerging Technologies and Factory Automation*, 2011, pp. 1–7.

[24] M. Ashjaei, M. Behnam, L. Almeida, and T. Nolte, "MTU configuration for real-time switched ethernet networks," *J. Syst. Architect.*, vol. 70, no. 4, pp. 15–25, Apr.

[25] K.-B. Gemlau, J. Peeck, N. Sperling, P. Hertha, and R. Ernst, "A new design for data-centric ethernet communication with tight synchronization requirements for automated vehicles," in *Proc. 45th Ann. Conf. IEEE Industrial Electronics Society*, 2019, pp. 4489–4494.

[26] N. Chaudhari, K. C. Ananthoju, and S. Isac, "Comparative analysis of large data transfer in automotive applications using Ethernet switched networks," in *Proc. Symp. Int. Automotive Technology*, 2019.

[27] B. Shin, J. Abdullayev, and D. Lee, "An efficient MAC layer packet fragmentation scheme with priority queuing for real-time video streaming," in *Proc. IEEE 41st Conf. Local Computer Networks*, 2016, pp. 69–77.

[28] J. Abdullayev, B. Shin, and D. Lee, "A dynamic packet fragmentation extension to high throughput WLANs for real-time H264/AVC video streaming," in *Proc. 10th Int. Conf. Future Internet*, 2015, pp. 1–4.

[29] I. Suciu, X. Vilajosana, and F. Adelantado, "An analysis of packet fragmentation impact in LPWAN," in *Proc. 2018 IEEE Wireless Communications and Networking Conf.*, 2018, pp. 1–6.

[30] S. A. Awwad, N. K. Noordin, B. M. Ali, F. Hashim, and N. H. A. Ismail, "6LoWPAN route-over with end-to-end fragmentation and reassembly using cross-layer adaptive backoff exponent," *Wirel. Pers. Commun.*, vol. 98, no. 1, pp. 1029–1053, 2018.

[31] X. Bao, Y. Zhang, D. Guo, and M. Song, "An optimization model for fragmentation-based routing in delay tolerant networks," *Science China Information Sciences*, vol. 59, no. 1.

[32] N. Naaman and R. Rom, "Packet scheduling with fragmentation," in *Proc. 21st Ann. Joint Conf. IEEE Computer and Communications Societies*, 2002, pp. 427–436.

[33] E. Suethanuwong, "Message fragmentation of event-triggered traffic in TTEthernet systems using the timely block method," in *Proc. Int. Conf. Computational Techniques in Information and Communication Technologies*, 2016, pp. 450–458.

[34] N. Semiconductors. (2017) Software user manual for SJA1105TEL. [Online]. Available: https://www.nxp.com/docs/en/user-guide/UM10944.pdf

[35] S. M. Laursen, P. Pop, and W. Steiner, "Routing optimization of AVB streams in TSN networks," *ACM Sigbed Review*, vol. 13, no. 4, pp. 43–48, 2016.

[36] V. Gavrilut, B. Zarrin, P. Pop, and S. Samii, "Fault-tolerant topology and routing synthesis for IEEE time-sensitive networking," in *Proc. 25th Int. Conf. Real-Time Networks and Systems*, 2017, pp. 267–276.

[37] P. Jayachandran and T. Abdelzaher, "A delay composition theorem for real-time pipelines," in *Proc. 19th Eur. Conf. Real-Time Systems*, 2007, pp. 29–38.

[38] A. Cimatti, A. Franzen, A. Griggio, R. Sebastiani, and C. Stenico, "Satisfiability modulo the theory of costs: foundations and applications," in *Proc. Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems*, 2010, pp. 99–113.

[39] E. B. Clark, T. A. Henzinger, H. Veith, and R. Bloem, "Satisfiability modulo theories," *Handbook of Model Checking*, pp. 305–343, 2018.

[40] J. Liu, *Real-time Systems*. Prentice Hall, 2000.

[41] N. C. Audsley, "On priority asignment in fixed priority scheduling," *Inf. Process. Lett.*, vol. 79, no. 1, pp. 39–44, 2001.

[42] A. Saifullah, Y. Xu, C. Lu, and Y. Chen, "End-to-end delay analysis for fixed priority scheduling in WirelessHART networks," in *Proc. 17th IEEE Real-Time and Embedded Technology and Applications Symp.*, 2011, pp. 13–22.

[43] N. Bjorner and A.-D. Phan, "vZ - maximal satisfaction with Z3," in *Proc. 6th Int. Symp. Symbolic Computation in Software Science*, 2014, pp. 1–9.

Xi Jin (M'17) received her Ph.D. degree in computer science from Northeastern University, China, in 2013. She is currently an Associate Professor with the Shenyang Institute of Automation, Chinese Academy of Sciences. She is also a committee member of the China Computer Federation Technical Committee on Embedded Systems (CCF TCEBS). Her research interests include industrial networks, real-time systems, and internet of things.

Changqing Xia (M'17) received the Ph. D. degree from Northeastern University, China in 2015. He is currently an associate professor at Shenyang Institute of Automation, Chinese Academy of Sciences. His research interests include wireless sensor networks, edge computing and real-time systems, especially the real-time scheduling algorithms, and smart energy systems.

Nan Guan is currently an assistant professor at the Department of Computing, The Hong Kong Polytechnic University. He received his BE and MS from Northeastern University, China in 2003 and 2006 respectively, and a PhD from Uppsala University, Sweden in 2013. His research interests include real-time embedded systems and cyber-physical systems. He received the EDAA Outstanding Dissertation Award in 2014, the Best Paper Award of RTSS 2009 and DATE 2013.

Peng Zeng received his Ph.D. degree from the Shenyang Institute of Automation, Chinese Academy of Sciences. He is currently a professor at the Shenyang Institute of Automation, Chinese Academy of Sciences. His research interests include industrial communication and wireless sensor networks.