An Intelligent Deterministic Scheduling Method for Ultralow Latency Communication in Edge Enabled Industrial Internet of Things

Yinzhi Lu, Liu Yang, Simon X. Yang, Senior Member, IEEE, Qiaozhi Hua, Arun Kumar Sangaiah, Member, IEEE, Tan Guo, Member, IEEE, and Keping Yu, Member, IEEE

Abstract—Edge enabled Industrial Internet of Things (IIoT) platform is of great significance to accelerate the development of smart industry. However, with the dramatic increase in real-time IIoT applications, it is a great challenge to support fast response time, low latency, and efficient bandwidth utilization. To address this issue, time sensitive network (TSN) is recently researched to realize low latency communication via deterministic scheduling. To the best of our knowledge, the combinability of multiple flows, which can significantly affect the scheduling performance, has never been systematically analyzed before. In this article, we first analyze the combinability problem. Then, a noncollision theory based deterministic scheduling (NDS) method is proposed to achieve ultralow latency communication for the time-sensitive flows. Moreover, to improve bandwidth utilization, a dynamic queue scheduling (DQS) method is presented for the best-effort flows. Experiment results demonstrate that NDS/DQS can well support deterministic ultralow latency services and guarantee efficient bandwidth utilization.

Index Terms—Edge intelligence, Industrial Internet of Things (IIoT), scheduling, smart industry, time sensitive network (TSN), ultralow latency communication.

I. INTRODUCTION

With the rapid development of Industrial Internet of Things (IIoT), artificial intelligence (AI), and edge computing, a new revolution of Industry 4.0 and beyond is now in progress, for the purpose of realizing smart industry [1]. With this trend, edge enabled IIoT platform is of great significance to accelerate the development of intelligent applications in industry, such as industrial surveillance, smart manufacturing, and predictive maintenance. However, due to the heterogeneity and limited computing and communication capabilities of edge nodes, the rapid increase of real-time IIoT applications make edge intelligence face a great challenge in terms of fast response, low latency, and efficient bandwidth utilization [2]. Therefore, how to satisfy rigorous requirements of IIoT systems is a significant problem to be solved.

Time sensitive network (TSN) is a promising technology that guarantees high real-time latency requirements for edge enabled IIoT [1], [3]. As an enhancement to Ethernet, TSN has recently attracted many researchers to explore its applications, for the purpose of assuring bounded low latency and jitters for time-sensitive flows and providing services for best-effort traffics [4]. Moreover, that TSN can help accelerating the revolution of IIoT to next generation operational efficiency and computing connectivity has reached a consensus. IEEE TSN Task Group has defined a whole range of 802.1 substandards, which involve the clock synchronization, frame preemption, network management, and scheduled traffic enhancement [5]. In particular, IEEE 802.1Qbv time aware shaper (TAS) has been known as a key technique to guarantee the deterministic low latency requirements via time-triggered communication that follows static schedules [4], [5].

In the past few years, some scheduling methods have been presented to get the global static schedules, with the aim of guaranteeing deterministic end-to-end latency for time-critical traffic in TSN [3], [6]. Queuing theory has been adopted to...
analyze the end-to-end transmission latency, and an analysis of the worst-case latency for time-sensitive flows has been made using network calculus [7], [8]. Whereas, one of the major drawbacks of the two methods is that the statistical analyzes of the transmission latency are based on the hypothetical arrival patterns of flows. Therefore, they are not suitable for latency analyzing in deterministic networks. In literatures [9] and [10], the scheduling of time-sensitive flows has been regarded as the Satisfiability/Optimization Modulo Theories (SMT/OMT) problem, which has been addressed by calculating the static schedules with an off-the-shelf solver. Although some suitable scheduling solutions may be acquired with SMT/OMT solvers, time complexity increases rapidly if the number of flows increases [11]. Scheduling with heuristic algorithms has been discussed in literatures [12] and [13], and some solutions with relatively lower time complexity have been presented.

Since the latency of flows in TSN depends not only on the scheduling, but also on the influence from other flows to be scheduled together. To address this issue, some researchers have focused on the joint optimization of flow scheduling and routing [14]–[16]. The flow route and schedule are jointly determined, whereas the updates are necessary when network structure changes or new flow occurs. Unfortunately, the delay caused by the conflicts between the time-sensitive flows in one port is rarely considered to date [17]. Moreover, current research studies only focus on the schedulability analysis for simple network topology, such as automotive in-vehicle network [18] and the network with tens of flows in industrial scenarios [19].

A. Motivation

With IIoT revolution towards next generation operational efficiency and computing connectivity, a large number of time-sensitive businesses need to be offloaded to the edge nodes for computing and analyzing. To support services with fast response time, low latency, and efficient use of bandwidth, the requirements of ultralow latency communication in edge enabled IIoT need to be satisfied. Then, how to rapidly acquire effective static schedules for massive time-sensitive flows is a major concern to be solved.

To address the above issue, in this article we analyze the scheduling problem based on the following two aspects: 1) to guarantee rigorous latency requirements of time-sensitive applications, all time-sensitive and best-effort flows must be strictly separated, while the performance requirements on delay and jitters must be satisfied. Existing methods designed based on SMT/OMT and integer linear programming (ILP) are with high complexity due to the missing of systematical analyzes on combinability for multiple flows. Then, it is a great challenge to optimize the scheduling for massive time-sensitive flows in edge enabled IIoT systems; 2) mandatory decentralized reservation of bandwidth for time-sensitive flows results in the fragmentation of residual time slots, and that a static schedule for time-sensitive flows divides the hyper-frame time slot into irregular fragments would reduce the utilization of remaining bandwidth. Therefore, an appropriate scheduling algorithm is necessary to guarantee the latency of time-sensitive flows and improve the overall bandwidth utilization for edge enabled IIoT systems.

B. Contribution

In this article, we mainly discuss the scheduling problem of time-sensitive applications in edge enabled IIoT, and our main contributions are given as follows.

1) Theoretical analyzes on the combinability problem of multiple flows are performed for the first time to our knowledge, and several conclusions about the noncollision combination of multiple time-sensitive flows are provided. Moreover, the conflicting packets between different flows can be predicted to make the scheduling scheme easy to be designed.

2) To satisfy bounded low latency and jitters requirements of time-sensitive flows, a deterministic scheduling method based on the combinability of multiple flows is presented. This method can intelligently determine whether nonconflict scheduling or the scheduling with relaxation of delay jitters is available.

3) To improve the overall bandwidth utilization, a dynamic scheduling method for nontime-sensitive flows is presented. The priorities, time slots, and queue lengths are considered together in this method, and the port revenue is maximized to determine the most appropriate packet to be forwarded.

The rest of this article is organized as follows. First, related works are summarized in Section II. Then, we describe the problems to be solved in Section III. Section IV gives the network model and theoretical analyzes. An overview and the details of the proposed scheduling methods are presented in Section V. Performance evaluations are performed for our methods in Section VI. Finally, Section VII, concludes the article.

II. RELATED WORKS

Recently, research of scheduling in TSN has been a major concern due to the rapid increase of time-sensitive businesses. Existing traffic scheduling methods in TSN can be divided into two categories: 1) the scheduling methods based on the Quality of Service (QoS); 2) the TAS-based scheduling methods [20]. Network calculus has been adopted to analyze the worst-case latency to guarantee the QoS of time-sensitive flows in literatures [7] and [8]. The scheduling problem has been analyzed by using the SMT/OMT model while off-the-shelf solvers have been introduced to calculate static TAS schedules in literatures [9], [10], [12], and [21].

Literature [22] has presented a method of time-sensitive software-defined network (SDN) to realize the scheduling for time-sensitive flows and best-effort flows in the same SDN. In addition, logical centralized controller has been adopted to address the global scheduling problem. ILP has been utilized to solve the routing and scheduling problems in literature [23], it enables the terminal to execute packet scheduling of the user plane with high precision. Heuristic algorithms have been used to optimize the scheduling in literatures [12], [13], [24], and
Table I

| Analysis | Literatures | [7] | [8] | [9] | [10] | [12] | [14] | [20] | [21] | [22] | [23] | [24] | [25] |
|----------|-------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| Focus    | Scheduling  | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
|          | Routing     | No  | No  | No  | No  | No  | Yes | Yes | No  | Yes | No  | No  | No  |
| Constraints | Constraints | No  | No  | No  | Yes | Yes | Yes | Yes | Yes | Yes | No  | No  | No  |
| Method   | SMT/OMT     | No  | No  | Yes | Yes | Yes | No  | No  | Yes | No  | No  | No  | No  |
|          | ILP         | No  | Yes | No  | No  | No  | Yes | Yes | No  | Yes | No  | Yes | No  |
|          | Network calculus | Yes | Yes | No  | No  | No  | No  | No  | No  | No  | No  | No  | No  |
| Optimization | Heuristic | No  | No  | No  | No  | Yes | No  | No  | Yes | No  | Yes | Yes | Yes |
| Application | Network size | Small | Small | Medium | Small | Medium | Small | Small | Small | Small | Small | Small |
|            | Scenarios   | Vehicle | Vehicle | IoT | IoT | IoT | IoT | Vehicle | IoT | IoT | IoT | IoT |
| Solver    | Off-the-shelf | No  | No  | Yes | Yes | No  | Yes | Yes | Yes | Yes | No  | No  | Yes |

[25], with the aim of minimizing the time complexity. Joint optimization of flow scheduling and routing has been explored in literatures [14] and [20], where the schedules of multiple flows in the routing nodes have been considered together to achieve deterministic end-to-end low latency.

Based on the reviews to existing works, the scheduling and routing of flows in TSN is either considered as a programming problem with constraints or an optimization problem with the aim of minimizing the end-to-end delay of flows, as shown in Table I. Although many existing works have focused on the delay guarantee with scheduling and routing algorithms, they lack theoretic analyzes on the key factors that result in the delay of TSN flows and do not further analyze the bandwidth utilization problem of best-effort flows. In contrast, we will theoretically analyze the combinability of time-sensitive flows to support the proposed scheduling methods, and consider the optimal combination of the strategies by using game theory to improve the overall bandwidth utilization.

### III. Problem Description

In edge enabled IIoT systems, multiple time-sensitive flows are inevitably scheduled and forwarded through the same node port during the data routing process. To achieve deterministic forwarding, some dedicated time slots are usually preserved in advance for the time-sensitive flows, while the residual is prepared for the best-effort ones. To optimize the scheduling of flows in edge enabled IIoT, the combinability of multiple time-sensitive flows and the residual bandwidth utilization (RBU) problems should be analyzed.

#### A. Combinability Problem of Time-Sensitive Flows

For the upcoming transformation in the context of Industry 4.0, many applications need end-to-end communications with hard-bounded guarantees for latency and reliability, as shown in Table II [4]. We call the data flows from those applications with bounded low latency and jitters as time-sensitive flows. However, fulfilling the requirement of one time-sensitive flow may make that of another flow unsatisfied. Hence, analyzes on combinability of multiple flows should be performed before the design of scheduling method.

To make the analyzes of the combinability problem easy to understand, we first give the definitions of some notations, as shown in Table III. The combinability of flows can directly affect the implementation of ideal scheduling. Generally, ideal deterministic scheduling for multiple time-sensitive flows in a node should satisfy two conditions: 1) the forwarding latency of flows is as low as possible; the original periodicity of flows is strictly maintained during the scheduling. Fig. 1 gives an...
example of ideal scheduling for two time-sensitive flows. It shows that the periods of the flows \( f_1 \) and \( f_2 \) in an egress port keep unchanged. For ideal scheduling, each packet in a flow must be ready before the reserved time slot arrives, while the latency should be as low as possible. Moreover, the reserved time slots for packets cannot conflict with each other in the time domain.

To describe the ideal scheduling problem for the flows \( f_1 \) and \( f_2 \), an optimization objective function can be defined as

\[
[\alpha_1, \alpha_2] = \arg \min_{\alpha_1, \alpha_2} (\alpha_1 - b_1, \alpha_2 - b_2)
\]

\[
\begin{align*}
C_1 & : \text{if} \ C_2 \text{ or } C_3, \\
C_1: & \tau_1 < (nT_1 + \alpha_1) - (mT_2 + \alpha_2) \\
C_2: & \tau_1 < (mT_2 + \alpha_2) - (nT_1 + \alpha_1), \quad \forall n, m \in N \\
C_3: & \begin{cases} \\
\alpha_1 - b_1 > J_1 + p_1 \\
\alpha_2 - b_2 > J_2 + p_2 \\
\end{cases}
\end{align*}
\]

(1)

where \( \alpha_1 \) and \( \alpha_2 \) denote the departure time of the 1st packets in the flows \( f_1 \) and \( f_2 \), respectively; \( b_1 \) and \( b_2 \) are the arrival time of the 1st packets; \( T_1 \) and \( T_2 \) are periods of \( f_1 \) and \( f_2 \), respectively; \( J_1 \) and \( J_2 \) are cumulated jitters before; \( p_1 \) and \( p_2 \) are the packet processing time of \( f_1 \) and \( f_2 \) in the switch node.

Ideal scheduling can effectively avoid delay jitters, because the \( n \)-th packet in the flow \( f_i \) is forwarded within the time slot \([nT_1+\alpha_1, nT_1+\alpha_1+\tau_1]\) so that all packets in the same flow have the same latency. However, whether multiple flows can be ideally scheduled together needs to be further analyzed since the conflicts between two flows may exist. The formal definition of the conflict is given as follows.

### Definition 1: Conflict between two flows \( f_1 \) and \( f_2 \) occurs if there exist two positive integers \((m, n)\) that make either of the following two conditions true.

1. \( nT_1 + \alpha_1 = mT_2 + \alpha_2 \);  
2. \( 0 < (nT_1 + \alpha_1) - (mT_2 + \alpha_2) < \tau_2 \) or \( 0 < (mT_2 + \alpha_2) - (nT_1 + \alpha_1) < \tau_1 \).

If the first condition is true, conflict of the first kind (CFK) occurs. If and only if the second is true, conflict of the second kind (CSK) appears. If neither CFK nor CSK occurs, multiple flows can be scheduled together to achieve ideal scheduling.

### B. Residual Bandwidth Utilization Problem

To guarantee deterministic delay, time slots for all packets in each time-sensitive flow should first be assigned. Then, the residual time slots can be shared by other best-effort flows. Whereas, this mandatorily allocation method inevitably results in the fragmentation of the residual time slots.

For traditional scheduling methods like strict priority (SP) and round robin, the remaining bandwidth is the only metric to measure whether flows are schedulable or not since it is continuously concentrated in the time domain. While the remaining bandwidth for deterministic scheduling is usually dispersed in the time domain. If the length of the packet to be scheduled does not match the size of a residual time slot, then this time slot may be wasted or partially occupied. Hence, to improve the remaining bandwidth utilization, it is necessary to take features of potential best-effort flows into consideration.

### IV. NETWORK MODEL AND THEORETICAL ANALYSES

In edge enabled IIoT, various time-sensitive applications such as multiagent communication, key-value collection, and remote real-time controlling increase rapidly. An edge enabled IIoT framework in time-sensitive scenarios is given in Fig. 2. For any time-sensitive application, the intelligent control layer first
provides the solutions for the centralized configuring, intelligent scheduling, and SDN based routing according to the corresponding requirements. Then, communication services are provided in edge layer [26], [27]. The data link layer finally performs packet delivering to guarantee latency requirements of the time-sensitive flows.

A. Network Model

A typical TSN network can be considered as the scheduling platform of time-sensitive applications, and it can be modeled as a graph $G = \{V, E\}$, where $V$ is the vertex set consisting of end nodes and switch nodes, and $E$ is the set of directed edges [10], [28]. Each directed edge denotes a physical link that connects two vertices.

An end-to-end unicast or multicast network flow from one end node $E_{\text{fa}}$ to another $E_{\text{fb}}$ via $n$ switch nodes ($SW_1 ... SW_n$), expressed as $f \rightarrow b$, can be considered as a tuple. A tuple $f \rightarrow b$ is denoted by a set $(f_{\rightarrow b}, R, f_{\rightarrow b}, \varphi, f_{\rightarrow b}, P)$, where $f_{\rightarrow b}, R$ is the directed subgraph $\{E_{\text{fa}} \rightarrow SW_1 \rightarrow SW_2 \rightarrow ... \rightarrow SW_n \rightarrow E_{\text{fb}}\}$ that describes the transmission path from $E_{\text{fa}}$ to $E_{\text{fb}}$ via $n$ switch nodes; $f_{\rightarrow b}, \varphi$ is the set of emerged time offsets when a flow passes through the egress ports of all switch nodes; $f_{\rightarrow b}, P$ is the set of all packets in the flow. For an intermediate switch node $SW_m$, the set of input flows is expressed as $\{f \rightarrow SW_m\}$, and the set of output flows is $\{SW_n.f \rightarrow \}$.

B. Theoretical Analyses on Combinability of Multiple Flows

The combinability means no conflict occurs for the flows transmitted via the same directed edge. While the conflict may be inevitable for the time-sensitive flows. To address this issue, we give some theorems below.

**Theorem 1:** For the flows $f_1$ and $f_2$, if their periods $T_1$ and $T_2$ are coprime ($\text{GCD}(T_1, T_2) = 1$), the conflict CFK occurs, and if $\tau_1 > 1$ or $\tau_2 > 1$, the conflict CSK occurs.

**Proof:** For any two flows $f_1$ and $f_2$, if $\text{GCD}(T_1, T_2) = 1$, then according to the Bezout’s theorem [29], $\exists n, m \in N$ make $nT_1 + o_1 = mT_2 + o_2$ true, therefore, CFK occurs. If $\text{GCD}(T_1, T_2) = 1$ and $\tau_1 > 1$, then $\exists n, m \in N$ make $(mT_2 + o_2) : (nT_1 + o_1) = 1$ true. That is, $0 < (mT_2 + o_2) : (nT_1 + o_1) < \tau_1$ is true and CSK occurs.

**Theorem 2:** For the flows $f_1$ and $f_2$, if CFK occurs, then the number pairs of the packets with CFK can be expressed as a set $(n, m)n = n_0 + kT_2/\text{GCD}(T_1, T_2), m = m_0 + kT_1/\text{GCD}(T_1, T_2)$, $k \in N$. Here $(n_0, m_0)$ is a particular number pair of the packets with CFK from $f_1$ and $f_2$.

**Proof:** Given two flows $f_1$ and $f_2$, if CFK occurs, $\exists n_0, m_0 \in N$ make the Bezout’s equation [30] $nT_1 + o_1 = mT_2 + o_2$ true. The complete solution of the equation can be achieved as $[n, m] = [n_0 + kT_2/\text{GCD}(T_1, T_2), m_0 + kT_1/\text{GCD}(T_1, T_2)]$. The integer solution space of $[T_1, T_2]$ consists of homogeneous general solution and nonhomogeneous particular solution, and it can be expressed as

$$X = k \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{K-1} \end{bmatrix} + \begin{bmatrix} x_{K-1}^0 \\ x_{K-1}^0 \\ \vdots \\ x_0 \\ x_0 \\ \vdots \\ x_0 \end{bmatrix}, k \in N. \quad (4)$$

A recursive method is adopted to solve (3). For the case that CFK between two flows ($K = 2$) occurs, the integer solution can

**Proof:** We try to prove the above lemma with the method of reduction to absurdity. Assuming that all the conditions above are satisfied for two flows with CFK or CSK. If CFK occurs, then $\exists n, m \in N$ make $nT_1 + o_1 = mT_2 + o_2$ true. Hence $nT_1 - mT_2 = o_2 - o_1$, and $nT_1 - mT_2 = kg$, so that $(o_1 - o_2)g = 0$. That is, if CFK occurs, the second condition is not satisfied. Similarly, it can be proven that if CSK occurs, the second condition cannot be satisfied. Therefore, the assumption is invalid.

**Theorem 3:** Noncollision combination of $K$ time-sensitive flows is available if all the following conditions are satisfied:

1) $g = \text{GCD}(T_1, ..., T_k) > 1$;
2) $\exists \tau_1 \leq (o_1 - o_2)g \leq g - \tau_1, \quad o_1 > o_2$;
3) $\tau_1 + \tau_2 < g$.

The proof of Theorem 3 is similar to Lemma 1. If both the first and the third conditions in Theorem 3 are satisfied, then there must exist a set $(o_1)$ that makes the second condition be satisfied.

**Theorem 4:** For $K$ time-sensitive flows, if $\text{GCD}(T_1, ..., T_K) = 1$, CFK occurs, and if $\exists \tau_i > 1 (i = 1, ..., K)$, CSK occurs. The proof of Theorem 4 is similar to Theorem 1.
be obtained based on Theorem 2, which is of the form
\[
X = k_2 \begin{bmatrix} T_2 \\ T_1 \end{bmatrix} + \begin{bmatrix} x_1^0 \\ x_2^0 \end{bmatrix}, \quad k_2 \in N.
\] (5)

The form of integer solution space of \(n\)-1 linear equations can be acquired based on (4). To search the solution of \(n\) linear equations, another equation is added that is expressed as follows:
\[
T_n x_n - T_{n+1} x_{n+1} = \alpha_{n+1} - \alpha_n.
\] (6)

The solution of (6) is
\[
\begin{bmatrix} x_n \\ x_{n+1} \end{bmatrix} = k_{n+1} \begin{bmatrix} T_{n+1} \\ T_n \end{bmatrix} + \begin{bmatrix} \rho_{n+1}^0 \\ \rho_n^0 \end{bmatrix}.
\] (7)

We can obtain \(x_n\) according to (4)
\[
x_n = k \alpha_n + x_0.
\] (8)

Then, we can easily get the solutions to \(k\) and \(k_{n+1}\) that can make both (7) and (8) be satisfied
\[
\begin{bmatrix} k \\ k_{n+1} \end{bmatrix} = k' \begin{bmatrix} T_{n+1}/GCD(a_n, T_{n+1}) \\ a_n/GCD(a_n, T_{n+1}) \end{bmatrix} + \begin{bmatrix} \rho_{n+1}^0 \\ \rho_n^0 \end{bmatrix}
\] (9)

where \(k' \in N\), \(\rho_{n+1}^0\) and \(\rho_n^0\) are nonhomogeneous particular solutions to the coefficient variables \(k\) and \(k_{n+1}\).

That \(a_n\) and \(T_{n+1}\) are coprime can also be proved using the recursive method. Then, (7) can be rewritten as
\[
\begin{bmatrix} x_n \\ x_{n+1} \end{bmatrix} = k' \begin{bmatrix} T_{n+1} a_n \\ T_n a_n \end{bmatrix} + \begin{bmatrix} a_n^0 \rho_{n+1}^0 + x_0 \rho_n^0 \\ T_n^0 x_{n+1}^0 + x_0^0 \end{bmatrix}.
\] (10)

Finally, we can get the integer solution space of (3)
\[
\begin{bmatrix} \Pi_{i \neq K-1} T_i \\ \Pi_{i \neq K} T_i \end{bmatrix} = k
\begin{bmatrix} \sum_{m=2}^{K-1} (\Pi_{j \neq 1, j \neq m} T_j)^0 x_{1m}^0 + \rho_{1m}^0 \\ \sum_{m=2}^{K-1} (\Pi_{j \neq 1, j \neq m} T_j)^0 x_{2m}^0 + \rho_{2m}^0 \\ \vdots \\ \sum_{m=2}^{K-1} (\Pi_{j \neq K-1, j \neq m} T_j)^0 x_{K-1m}^0 + \rho_{K-1m}^0 \\ \sum_{m=2}^{K-1} (\Pi_{j \neq K} T_j)^0 x_{Km}^0 + \rho_{Km}^0 \end{bmatrix}
\] (11)

where \(x_{im}^0\) and \(\rho_{im}\) (\(i = 1, 2, \ldots, K\)) are the particular solutions of the \(i\)th equation in (2) and the related coefficient variable \(k_m\) in the \(m\)th recursive calculation, respectively.

The worst combination of \(K\) time-sensitive flows is that the periods of any two flows are coprime, and the CFK among \(K\) flows occurs every \(\Pi T_i\).

Similarly, if there exists CSK among \(K\) flows, (12) has integer solution space
\[
\begin{align*}
T_1 x_1 - T_2 x_2 &= o_2 - o_1 + v_1 \\
T_2 x_2 - T_3 x_3 &= o_3 - o_2 + v_2 \\
&\vdots \\
T_{K-1} x_{K-1} - T_K x_K &= o_K - o_{K-1} + v_{K-1}
\end{align*}
\] (12)

where \(v_i\) (\(i = 1, \ldots, K-1\)) denotes the overlap length in time domain. It should be noted that, here \(v_i\) should be nonzero, if not, it means that CFK exists. We can get the solution space of (12) using the similar method adopted for (2).

To make it easy to understand, an example is given here to explain how to get the solution. Assuming that there are three flows to be scheduled, and their periods are coprime so that CFK exists. The flow parameters are shown in Table IV. We can get the linear equations about the worst combination:
\[
\begin{align*}
f_1 &= 14 x_1 - 27 x_2 = 5 - 0 \\
f_2 &= 27 x_2 - 61 x_3 = 9 - 5
\end{align*}
\] (13)

The integer solution of \(14 x_1 - 27 x_2 = 5\) is as follows:
\[
\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = k_1 \begin{bmatrix} 27 \\ 14 \end{bmatrix} + \begin{bmatrix} 10 \\ 5 \end{bmatrix}, \quad k_1 \in N.
\] (14)

Similarly, we can get the integer solution of \(27 x_2 - 61 x_3 = 4\) as follows:
\[
\begin{bmatrix} x_2 \\ x_3 \end{bmatrix} = k_2 \begin{bmatrix} 61 \\ 27 \end{bmatrix} + \begin{bmatrix} 25 \\ 11 \end{bmatrix}, \quad k_2 \in N.
\] (15)

Then, we can easily get the solutions to \(k_1\) and \(k_2\) that can make (14) match (15)
\[
\begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = k \begin{bmatrix} 61 \\ 14 \end{bmatrix} + \begin{bmatrix} 45 \\ 10 \end{bmatrix}, \quad k = 1, 2, 3, \ldots
\] (16)

Therefore, the solution space of (13) is
\[
\begin{align*}
x_1 &= k_1 \begin{bmatrix} 27 \cdot 61 \\ 14 \cdot 61 \end{bmatrix} + \begin{bmatrix} 14 \cdot 45 + 10 \\ 27 \cdot 10 + 11 \end{bmatrix}, \quad k = 0, 1, 2, \ldots
\end{align*}
\] (17)

Based on the results, we can confirm the conflicting packets. For example, if \(k = 0\), CFK occurs for the packets numbered \([27+45+10, 14+45+5, 27+10+11]\) from the three flows.

To the best of our knowledge, this is the first time to make theoretical analyzes for combinability of \(K\) time-sensitive flows, which will guide the scheduling design and provide theoretical basis. The conditions of noncollision combination for flows are provided in Theorem 3. For multiple flows to be scheduled together, we first consider whether these conditions are true. If they are, jitters-free scheduling may be available for the flows. Otherwise, that all conflicting packets can be confirmed in advance makes it easy to eliminate future conflicts during the scheduling.
otherwise, the payers consisting of game model data forwarding model for switch nodes to make decision. A dynamic scheduling method needs to be presented. We give a ones. To assign the two kinds of services into a schedule, a should be imposed on the corresponding switch node. Third, if any packet is dropped in a port, the punishment time-sensitive flows should not be disturbed by the best-effort than that of transmitting a best-effort packet. Therefore, the a time-sensitive packet in the reserved time slot is much higher of maximizing the revenue. Second, the benefit of transmitting switch nodes are rational and selfish individuals with the aim hypotheses and preconditions. First, we assume that all the

### A. Data Forwarding Model

Before giving the scheduling model, let us present some hypotheses and preconditions. First, we assume that all the switch nodes are rational and selfish individuals with the aim of maximizing the revenue. Second, the benefit of transmitting a time-sensitive packet in the reserved time slot is much higher than that of transmitting a best-effort packet. Therefore, the time-sensitive flows should not be disturbed by the best-effort ones. Third, if any packet is dropped in a port, the punishment should be imposed on the corresponding switch node.

Deterministic delay services usually coexist with best-effort ones. To assign the two kinds of services into a schedule, a dynamic scheduling method needs to be presented. We give a data forwarding model for switch nodes to make decision. A game model \( G = (P, S, U) \) is formally defined. Where \( P \) is the payers consisting of \( n \) queues in an egress port; \( S = \{s_i\} \) is the strategy set; and \( U = \{u_i\} \) is the utility function set. It is assumed that, as the player, each port is a rational and selfish intelligent individual [31]. Then, each player always expects to improve its own payoff, which not only depends on its own strategy, but also subjects to the states of others.

For an egress port, a packet from one queue will be chosen for transfer if it is ready while the port is free. The strategy determined by the queue is a finite pure strategy that is either “ready for transfer” or “not ready.” If a packet in a queue \( i \) is selected to be ready for transfer, then the strategy is \( s_i = 1 \); otherwise, \( s_i = 0 \). The strategy subjects to \( \sum s_i \leq 1 \), \( \forall s_i = 0 \) or 1.

Each queue in an egress port ends with a time-aware gate controlled by gate control list (GCL) in TSN [26], as shown in Fig. 3. Every row in GCL contains some gate sates that are either open or close [8]. From the queues whose gates are open, packets are selected sequentially and ready for transfer based on the designed scheduling method. We try to give a dynamic transmission selection algorithm to keep a better coexistence for the time-sensitive and best-effort flows.

### V. PROPOSED SCHEDULING METHOD

#### A. Data Forwarding Model

Before giving the scheduling model, let us present some hypotheses and preconditions. First, we assume that all the switch nodes are rational and selfish individuals with the aim of maximizing the revenue. Second, the benefit of transmitting a time-sensitive packet in the reserved time slot is much higher than that of transmitting a best-effort packet. Therefore, the time-sensitive packets are selected sequentially and ready for transfer based on the states of others. Whether a packet is selected or not depends on its own strategy, but also subjects to the states of others.

For an egress port, a packet from one queue will be chosen for transfer if it is ready while the port is free. The strategy determined by the queue is a finite pure strategy that is either “ready for transfer” or “not ready.” If a packet in a queue \( i \) is selected to be ready for transfer, then the strategy is \( s_i = 1 \); otherwise, \( s_i = 0 \). The strategy subjects to \( \sum s_i \leq 1 \), \( \forall s_i = 0 \) or 1.

Each queue in an egress port ends with a time-aware gate controlled by gate control list (GCL) in TSN [26], as shown in Fig. 3. Every row in GCL contains some gate sates that are either open or close [8]. From the queues whose gates are open, packets are selected sequentially and ready for transfer based on the designed scheduling method. We try to give a dynamic transmission selection algorithm to keep a better coexistence for the time-sensitive and best-effort flows.

### B. Noncollision Deterministic Scheduling

Time-sensitive flows always have the strict delay and jitters constraints. Then, deterministic scheduling methods need to be designed for the purpose of providing services with bounded low latency and low jitters for the flows. The assignment of time slots for time-sensitive flows is crucial to satisfy the delay requirements. To address this issue, we present a noncollision deterministic scheduling (NDS) method for TSN flows.

An overview of the proposed NDS method is given in Fig. 4. Since bandwidth constraint must be satisfied for deterministic scheduling, the total bandwidth occupied by all time-sensitive flows cannot exceed the bandwidth of a directed edge. If the greatest common divisor \( g \) is greater than 1, while the total service time for \( K \) packets, which are separately selected from \( K \) time-sensitive flows, is less than \( g \), then nonconflict ideal scheduling method can be considered. Otherwise, scheduling method with relaxation of delay jitters is adopted.

For a given flow set \( F \) that contains \( K \) time-sensitive flows, the static schedule calculation method is given in Algorithm 1. If conditions 1 and 3 in Theorem 3 are satisfied, nonconflict ideal scheduling method can be adopted. Then, the time-offset set \( O \) needs to be determined using the presented method in Algorithm 2. Based on the acquired set \( O \), the static schedules can be directly computed. For example, when the \( n \)th packet of the flow \( f \) in a queue is ready for transfer, the gate of the queue will be open within the time slot \([nT_1+o_1, nT_2+o_2+\tau]\). Otherwise, scheduling method with relaxation of delay jitters has to be adopted. First, the flow set \( F \) needs to be divided into several subsets. Then, to restrict the occurrence of CFK and CSK, nonconflict ideal scheduling is used for every subset if all applicable conditions are true. Next, to eliminate conflicts between flows, a searching method is adopted to perform relaxation operation of the constrained delay jitters. The static schedule within a cycle that is the least common multiple of periods for the flows in set \( F \) is determined.

For a flow set \( F \), it needs to be divided into several subsets if either CFK or CSK is inevitable. The flows that satisfy the
Algorithm 1: Static Schedule Calculation for $K$ Flows.

**Input:** Set $F$ of $K$ flows, initial time-offset set $\varphi$, delay-jitters constraint set $C$, processing time set $P$

**Output:** The static schedule $SCH$

1: if $GCD(F) > 1 \land \sum_{j} \tau_j \leq g$ then
2: $(O, S) \leftarrow$ Non-conflict calculation($F$, $\varphi$, $C$, $P$);
3: $SCH \leftarrow \{nT_i + o_i, nT_i + o_i + \tau_i\}$;
4: else
5: $\{F_1, F_2, ..., F_m\} \leftarrow$ set-partition($F$);
6: $\{g_1, g_2, ..., g_m\} \leftarrow \{GCD(F_i), |i = 1, 2, ..., m|\}$;
7: Initialize $O$;
8: for-each $F_i$ (from 1 to $m$) if $g_i > 1 \land \sum \tau_j \leq g_i$ then in remaining time slots
9: $(O_i, S) \leftarrow$ Non-conflict calculation($F_i$, $\varphi_i$, $C_i$, $P_i$);
10: end if
11: end for-each
12: Loop to search $SCH$ until $C$ is satisfied or time out for all packets within the least common multiple (LCM) of periods for flows in $F$
13: if CFK or CSK occurs then
14: Eliminate the conflicts based on $C$;
15: end if
16: end Loop
17: end if
18: if $S = 0$ or $C$ is satisfied then
19: Output the $SCH$;
20: end if
21: end if

combimability can be grouped into the same subset. Moreover, each subset from $F$ expects to contain as many combinative flows as possible. Then, the time complexity of the scheduling can be further reduced while the requirements for delay-jitters are easier to be satisfied.

C. Dynamic Queue Scheduling

The above method NDS aims at guaranteeing the latency of time-sensitive flows through analyzing the combinability of multiple flows. Whereas, mandatory allocation of bandwidth for time-sensitive flows inevitably results in the fragmentation of the residual time slots. To improve the overall bandwidth utilization, other best-effort flows with QoS requirements need to be elaborately scheduled within the residual time slots. To address this issue, we present a dynamic queue scheduling (DQS) method for those nontime-sensitive flows.

For the scheduling of nontime-sensitive flows, the strategy of an egress port not only depends on the present available time slots and the number of packets that are ready for transfer in the queues, but also is related to the strategies of the ports in neighbor nodes. In addition, the number of packets that are ready for transfer in a queue may be affected by the present strategy and the number of packets arrived from the neighbors in the followed period. To address this issue, reinforcement learning is borrowed to design the dynamic queue scheduling.

Algorithm 2: Non-Conflict Time-Offset Set Calculation.

**Input:** $F$, $\varphi$, $C$, $P$

**Output:** Time-offset set $O$ of $F$, unsolved identifier $S$

1: $g \leftarrow GCD(T_1, T_2, ..., T_n)$;
2: $l \leftarrow LCM(T_1, T_2, ..., T_n)$;
3: $o_i \leftarrow \varphi_i + p_i$;
4: for-each $o_i \in O$
5: while $(o_i < T_i)$
6: if for all $o_j \ (j \neq i) \ (k-1)g + \tau_i \leq o_i - o_j \leq kg - \tau_i$;
7: stop;
8: else
9: $o_i++$;
10: if $o_i == c_i$ then
11: $S = 1$;
12: end if
13: end if
14: end while
15: end for-each
16: if $S = 1$ then
17: $O \leftarrow \{o_1, o_2, ..., o_K\}$;
18: end if
19: return $O, S$

For scheduling, a port always expects to maximize its own payoff that may be affected by the present and next strategies. Therefore, the present and next strategies should be considered when designing utility functions. The present utility function for the strategy $s$ is first given as follows:

$$u^{pr} = c \cdot s^T - \beta p \cdot (ones(1, n) - s^T)$$  \hspace{1cm} (18)

where $c = [c_1 \ ... \ c_n]$ is the beneficial coefficient vector, and $p = [p^0 \ p^1 \ ... \ p^K]$ is the penalty factor vector for packet loss.

Penalty factor vector $p$ depends on the number of packets in queues, then penalty factor $p^i$ regarding queue $i$ is of the form

$$p^i = \begin{cases}
0 & q_{L}^i / q_{L}^{max} \geq \frac{q_{L}^{max}}{2} \\
\frac{q_{L}^{max}}{2} & q_{L}^{max} / 2 \leq q_{L}^{max} < q_{L}^{max} \\
q_{L}^{max} & q_{L}^{max} \leq q_{L}^{max} / q_{L}^{max}
\end{cases}$$  \hspace{1cm} (19)

where $q_{L}^{i}$ is the number of packets ready for transfer, $q_{L}^{max}$ and $p_{0}$ are the maximum queue length and penalty, respectively.

The utility function of next strategy $\bar{s}$ can be expressed as

$$u^{ns} = c \cdot s^T - \beta (\bar{p} - s) \cdot (ones(1, n) - s^T)$$  \hspace{1cm} (20)

where $\bar{p}$ is the predicted penalty factor vector, and the $i$th factor is of the form

$$\bar{p}^i = \begin{cases}
0 & q_{L}^{max} / q_{L}^{max} \geq q_{L}^i + \tau E_a^T \\
\frac{q_{L}^{max}}{2} & q_{L}^{max} / 2 \leq q_{L}^i + \tau E_a^T < q_{L}^{max} \\
q_{L}^{max} & q_{L}^{max} \leq q_{L}^i + \tau E_a^T
\end{cases}$$  \hspace{1cm} (21)

where $E_a(t)$ is the expected number of the arrival packets in time slot $t$, and $E_a^T$ is the expected number within the period $T$.

The final utility function of the port is a combination of the above two utility functions, which can be expressed as

$$u = \alpha u^{pr} + (1 - \alpha) u^{ns}.$$  \hspace{1cm} (22)
Hence, the best strategy $s^*$ of a port can be obtained by maximizing its utility

$$s^* = \arg \max_s (u).$$  \hspace{1cm} (23)\

For our scheduling methods NDS/DQS, the deterministic delay of time-sensitive flows can be guaranteed using the NDS method. Lower priorities are assigned to the best-effort flows, which are scheduled in the remaining decentralized time slots. Both the fragmentation of the remaining time slots and the dynamic characteristic regarding the number of ready packets in queues are considered in the DQS method. Services for the best-effort flows are expected to be guaranteed while the utilization of the remaining bandwidth can be improved.

### VI. SIMULATION RESULTS AND ANALYSIS

In this section, we perform the experiments to evaluate the performance of our scheduling algorithms. Experiments run on an Intel(R) Core(TM) i5-2600 64 bit CPU@3.4 GHz with 12 GB of RAM via MATLAB R2020b.

#### A. Simulation Setup

In the simulations, we evaluate the delay and jitters of time-sensitive flows and the bandwidth utilization of best effort flows in a switch node. All egress ports are full-duplex and with 1 Gbit/s bandwidth.

We refer to the descriptions of flow properties in IEC/IEEE 60802 to simulate the flows in actual industrial scenarios. For time-sensitive flows, the packet sizes range from 64 to 512 B, the periods are 0.5 ms, 2 ms or 5 ms, the arrival time of packets is randomly generated. To simulate the flows under different industrial scenarios and evaluate the robustness of algorithms, we count the delay and jitters for 5 to 100 flows on a single directed edge, as shown in Table V, and the ratio of flows with different periods is nearly 1:1:1.

#### B. Performance of NDS Method

To better evaluate the proposed flow scheduling algorithms, we focus on the forwarding delay for a single switch node and the flow scheduling on a single directed edge. Assuming that there are $m$ flows to be forwarded in a switch node $SW_i$ and $n$ flows need to be scheduled on the directed edge $SW_i \rightarrow SW_j$.

The results of delay for all flows scheduled on $SW_i \rightarrow SW_j$ using the proposed algorithm NDS and the comparison algorithms dynamic mini-max (D/MM), DoC-aware iterative routing and scheduling (DA/IRS), and efficient probing instructed by conflicts (EPIC) in literatures [11], [14], and [28] are shown in Fig. 5. To verify the performance of our proposed algorithm, we count the maximum, minimum, and average delay of 20 time-sensitive flows assigned to one directed edge. We use error bar to show deterministic delay of flows for D/MM and DA/IRS. These flows are approximately divided into three even groups with the periods 0.5 ms, 2 ms, and 5 ms. The packet sizes of these flows range from 64 to 512. Compared with the other three algorithms, NDS can provide nonconflict and jitters-free scheduling for flows as expected. Because the total service time for the 20 packets, which are separately selected from the above 20 time-sensitive flows, is less than the greatest common divisor of periods (0.5 ms, 2 ms, and 5 ms), then the first and third conditions in Theorem 3 are satisfied. Hence, it is certain that a time-offset set $\{o_i\}$ exists to make the second condition satisfied. The forwarding delay of each packet from the 20 flows is counted in the experiment. The results for the maximum, minimum, and average delay of every flow are shown in this figure. The EPIC algorithm can also achieve the jitters-free scheduling for flows. However, the delay using EPIC is much higher than NDS. Jitters of the 20 flows using D/MM and DA/IRS are relatively higher than the average delay. This indicates that the two algorithms have a weaker performance in guaranteeing deterministic forwarding services than NDS and EPIC, since the jitter of a flow is the difference between the maximum and minimum delay. The delay using NDS is lower than the delay using EPIC and the average delay using D/MM and DA/IRS, this means that the noncollision combination theory can better guide scheduling optimization.

Taking the machine control as an example, the commands should be transmitted to the destination within 0.5 ms while jitters should be avoided. Fig. 5 shows jitters-free scheduling can be achieved using the proposed algorithm NDS, and the delay is no more than 15 us for 20 flows scheduled together in a node. Therefore, algorithm NDS can meet the requirements of the time-sensitive applications in intelligent manufacturing, which is essential for edge enabled IIoT.

The comparison of execution time for these algorithms is given in Fig. 6. Here the number of time-sensitive flows ranges from 5 to 100. It can be noticed that, with the increase of the number of flows, the execution time of D/MM, DA/IRS, and EPIC increases faster than that of NDS. The results also

| Flows number | Periods $T$ | Ratios | Packet sizes |
|--------------|-------------|--------|--------------|
| 5            | (0.5ms, 2ms, and 5ms) | 2:2:1  | 64 to 512B   |
| 20           | (0.5ms, 2ms, and 5ms) | 6:7:7  | 64 to 512B   |
| 50           | (0.5ms, 2ms, and 5ms) | 16:17:17 | 64 to 512B   |
| 100          | (0.5ms, 2ms, and 5ms) | 33:33:34 | 64 to 512B   |
show that the average execution time of NDS for 100 flows is 33.7 ms, while those of EPIC, D/MM, and DA/IRS are 98.6 ms, 165.5 ms, and 428.6 ms, respectively. This big gap owes to that D/MM and DA/IRS search the qualified solutions for all packets in a superperiod (the least common multiple of periods for total 100 flows) without/with delay and jitters constraints, and EPIC searches the solutions by seeking for time offsets to make any two flows noncollision. While NDS only computes these appropriate time offsets of flows so that the complexity is greatly simplified and the execution time can be reduced.

The comparison of schedulable ratios among NDS, D/MM, DA/IRS, and EPIC is shown in Fig. 7. It shows that, the schedulable ratio of NDS keeps high while those of D/MM, DA/IRS, and EPIC decrease with the increase of the number of flows. In our proposed algorithm NDS, the combinability of multiple flows is analyzed. When all conditions in Theorem 3 are satisfied, the schedule can be computed only by searching for the optimal time-offset set. Otherwise, relaxation of jitters needs to be considered to resolve the conflicts. Then, packets with CFK or CSK can be confirmed via theoretical deduction, so that the optimal solutions of scheduling are easy to search. This figure also shows that D/MM has the lowest schedulable ratio for all cases since it ignores delay and jitters constraints, which are considered by all other algorithms.

C. Performance of DQS Method

To improve the overall bandwidth utilization, a scheduling method DQS for the best-effort flows is also presented in this article. In our proposed method, the present and next strategies are considered together to maximize the bandwidth utilization. The weight coefficients of the benefits for the present and next strategies are both set as 0.5. To evaluate the performance, experiment is performed under the case that the number of time-sensitive flows ranges from 5 to 100 while the percent of bandwidth required for the best-effort flows is nearly 50.

The comparison of bandwidth utilization is given in Fig. 8. It shows that SP has the highest bandwidth utilization, while the gap of bandwidth utilization between SP and the other three algorithms increases with the increase of the number of time-sensitive flows. This figure also shows that NDS/DQS has higher bandwidth utilization than D/MM, DA/IRS, and EPIC. This is because the present and next strategies are jointly considered in NDS/DQS to reduce the number of idle time slots. To some extent, the proposed method can reduce the ratio of packet loss caused by queuing, especially for the best-effort flows with low priorities and long packets. Hence, the proposed method NDS/DQS can help to realize ultralow latency communication in TSN networks.

VII. CONCLUSION

To support the services with fast response time, low latency, and efficient use of bandwidth in edge enabled IIoT, in this article, we presented the traffic scheduling method NDS/DQS to guarantee the ultralow latency communication. The proposed method includes deterministic scheduling algorithm NDS for the time-sensitive flows and queue scheduling algorithm DQS for the best-effort flows. The combinability of time-sensitive flows was analyzed and formalized in NDS, while the RBU problem is concerned in DQS. Experiments are performed to verify the performances regarding delay and jitters, execution time, schedulable ratio, and bandwidth utilization. The results demonstrate the effectiveness and outperformance of the proposed method in scheduling for both time-sensitive and best-effort flows.
In future work, we will further research ultralow latency communication and study the deterministic latency guarantee techniques for IoT. The applications of AI in scheduling will be explored to optimize the delay and jitters for time-sensitive flows. To ensure the end-to-end latency, deterministic routing protocols based on the hypergraph model will be our research focus.

REFERENCES

[1] Y. Wang, S. Yang, X. Ren, P. Zhao, C. Zhao, and X. Yang, “IndustEdge: A time-sensitive networking enabled edge-cloud collaborative intelligent platform for smart industry,” IEEE Trans. Ind. Informat., vol. 18, no. 4, pp. 2386–2398, Apr. 2022.

[2] C. Avasaki, B. Zarrin, and S. Dustdar, “Edge flow -developing and deploying latency-sensitive IoT edge applications,” IEEE Internet Things J., vol. 9, no. 5, pp. 3877–3888, Mar. 2022, doi: 10.1109/JIOT.2021.3101449.

[3] A. Nasrallah et al., “Ultra-low latency (ULL) networks: The IEEE TSN and IETF Detnet standards and related 5G ULL research,” IEEE Commun. Surv. Tut., vol. 21, no. 1, pp. 88–145, Firstquarter 2019.

[4] S. Bigo, N. Benzaoui, K. Christodouloupolous, R. Miller, W. Lauten-schlaeger, and F. Frick, “Dynamic deterministic digital infrastructure for time-sensitive applications in factory floors,” IEEE J. Sel. Topics Quantum Electron., vol. 18, no. 6, Nov./Dec. 2012, doi: 10.1109/JQLE.2012.2221080.

[5] W. Steiner, S. S. Craciunas, and R. S. Oliver, “Traffic planning for time sensitive communication,” IEEE Commun. Standards Mag., vol. 2, no. 2, pp. 42–47, Jun. 2018.

[6] S. S. Craciunas and R. S. Oliver, “An overview of scheduling mechanisms for time-sensitive networks.” Real-Time Summer School, L’Ecole de Tem ples, Paris, France, Rep. 2017.

[7] L. Zhao, P. Pop, and S. S. Craciunas, “Worst-case latency analysis for IEEE 802.1Qbv time sensitive networks using network calculus,” IEEE Access, vol. 6, pp. 41803–41815, Jul. 2018.

[8] J. Diemer, D. Thiele, and R. Ernst, “Formal worst-case timing analysis of Ethernet topologies with strict-priority and AVB switching,” in Proc. Int. Symp. Ind. Embedded Syst., Jun. 2012, pp. 187–192.

[9] F. Pozo, W. Steiner, G. Rodriguez-Navas, and H. Hanson, “A decomposition approach for SMF-based schedule synthesis for time-triggered networks,” in Proc. IEEE 20th Conf. Emerg. Technol. Factory Automat., 2015, pp. 1–8.

[10] S. S. Craciunas, R. S. Oliver, M. Chmelik, and W. Steiner, “Scheduling real-time communication in IEEE 802.1 Qbv time sensitive networks,” in Proc. ACM Int. Conf. Real Time Netw., Brest, France, 2016, pp. 183–192.

[11] A. A. Atallah, G. B. Hamad, and O. A. Mohamed, “Routing and scheduling of time-triggered traffic in time-sensitive networks,” IEEE Trans. Ind. Informat., vol. 16, no. 7, pp. 4525–4534, Jul. 2020.

[12] X. Jin et al., “Real-time scheduling of massive data in time sensitive networks with a limited number of schedule entries,” IEEE Access, vol. 8, pp. 6751–6767, Jan. 2020.

[13] M. L. Raag aard, P. Pop, M. Gutierrez, and W. Steiner, “Runtime reconguration of time-sensitive networking (TSN) schedules for fog computing,” in Proc. IEEE Fog World Cong., Oct. 2017, pp. 1–6.

[14] N. G. Nayak, F. Diir, and K. Rothermel, “Incremental flow scheduling in time-sensitive networking flows in automotive communications,” IEEE Commun. Lett., vol. 14, no. 5, pp. 2066–2075, May 2018.

[15] A. A. Syed, S. Ayaz, T. Leinmüller, and M. Chandra, “Dynamic scheduling and routing for TSN based in-vehicle networks,” in Proc. IEEE Int. Conf. Commun. Workshops, 2021, pp. 1–6.

[16] F. Smirnov, M. Gläss, F. Reimann, and J. Teich, “Optimizing message routing and scheduling in automotive mixed-criticality time-triggered networks,” in Proc. IEEE/ACM/EDAC 54th Des. Autom. Conf., Jun. 2017, pp. 1–6.

[17] K. Huang et al., “A period-aware routing method for IEEE 802.1Qbv TSN networks,” Electronics, vol. 10, pp. 1–19, Dec. 2020.

[18] L. Leonard, L. L. Bello, and G. Patti, “Bandwidth partitioning for time-sensitive networking flows in automotive communications,” IEEE Commun. Lett., vol. 25, no. 10, pp. 3258–3261, Oct. 2021.

[19] L. Lo Bello, M. Ashjaei, G. Patti, and M. Behnam, “Schedulability analysis of time-sensitive networks with scheduled traffic and pre-emption support,” J. Parallel Distrib. Comput., vol. 144, pp. 153–171, Oct. 2020.

[20] F. Smirnov, M. Gläss, F. Reimann, and J. Teich, “Optimizing message routing and scheduling in automotive mixed-criticality time-triggered networks,” in Proc. IEEE/ACM/EDAC 54th Des. Autom. Conf., Jun. 2017, pp. 1–6.

[21] R. S. Oliver, S. S. Craciunas, and W. Steiner, “IEEE 802.1Qbv gate control list synthesis using array theory encoding,” in Proc. IEEE Real Time Embedded Technol. Appl. Symp., Porto, Portugal, Apr. 2018, pp. 13–24.

[22] N. G. Nayak, F. Diir, and K. Rothermel, “Time-sensitive software-defined network (TSSDN) for real-time applications,” in Proc. 24th Int. Conf. Real-Time Netw. Syst., pp. 193–202, 2016.

[23] P. Pop, M. L. Raagaard, S. S. Craciunas, and W. Steiner, “Design optimisation of cyber-physical distributed systems using IEEE time sensitive networks,” IET Cyber Phys. Syst. Theory Appl., vol. 1, no. 1, pp. 86–94, Dec. 2016.

[24] G. Gavrilut and P. Pop, “Scheduling in time sensitive networks (TSN) for mixed-criticality industrial applications,” in Proc. IEEE 14th Int. Workshop Factory Commun. Syst., Jun. 2018, pp. 1–4.

[25] F. Diir and N. G. Nayak, “No-wait packet scheduling for IEEE time sensitive networks (TSN),” in Proc. ACM Int. Conf. Real Time Netw. Syst., Brest, France, 2016, pp. 203–212.

[26] L. Yang, K. Yu, S. X. Yang, C. Chakraborty, Y. Lu, and T. Guo, “An intelligent trust cloud management method for secure clustering in 5G enabled internet of medical things,” IEEE Trans. Ind. Informat., to be published, 2021, doi: 10.1109/TII.2021.3128954.

[27] B. Zhu, K. Chi, J. Liu, K. Yu, and S. Mutmaz, “Efficient offloading for minimizing task computation delay of NOMA-based multi-access edge computing,” IEEE Trans. Commun., vol. 70, no. 5, pp. 3186–3203, May 2022, doi: 10.1109/TCOMM.2022.316263.

[28] M. Vik, K. Brejchová, Z. Hanáček, and S. Tang, “Large-scale periodic scheduling in time-sensitive networks,” Comput. Oper. Res., vol. 137, 2022, Art. no. 105512, doi: 10.1016/j.cor.2021.105512.

[29] G. L. Feng, T. R. N. Rao, G. A. Berg, and J. Zhu, “Generalized Bezout’s theorem and its applications in coding theory,” IEEE Trans. Inform. Theory, vol. 43, no. 6, pp. 1799–1810, Nov. 1997.

[30] H. Boche and V. Pohl, “General characterization of Bezout space-time equalizers and precoders,” in Proc. Int. Zurich Seminar Commun., Zurich, Switzerland, Feb. 2006, pp. 30–33.

[31] L. Yang, Y. Z. Lu, Y. C. Zhong, X. G. Wu, and S. J. Xing, “A hybrid, game theory based, and distributed clustering protocol for wireless sensor networks,” Wireless Netw., vol. 22, no. 3, pp. 1007–1021, Apr. 2016.

Yinzhi Lu received the M.S. degree in communication and information systems from Chongqing University, Chongqing, China, in 2014. She is currently working toward the Ph.D. degree in information and communication engineering with the School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing, China. From 2014 to 2019, she was a Teaching Assistant with the School of Electronic Information Engineering, Yangtze Normal University, Chongqing, China. Her current research interests include Internet of Things, time sensitive network, and artificial intelligence.

Liu Yang received the B.S. degree in electronic information science and technology from the Qingdao University of Technology, Shandong, China, and Ph.D. degree in communication and information systems from the School of Communication Engineering, Chongqing University, Chongqing, China, in 2010 and 2016, respectively. He is currently a Lecturer with Chongqing University of Posts and Telecommunications, Chongqing, China. His research interests include Internet of Things, data analysis, and artificial intelligence.
Simon X. Yang (Senior Member, IEEE) received the B.Sc. degree in engineering physics from Beijing University, Beijing, China, the M.Sc. degree in biophysics from the Chinese Academy of Sciences, Beijing, China, and the Ph.D. degree in electrical engineering from the University of Houston, Houston, TX, USA, and the Ph.D. degree in electrical and computer engineering from the University of Alberta, Edmonton, AB, Canada, in 1987, 1990, 1996, and 1999, respectively.

He is currently a Professor and the Head of the Advanced Robotics and Intelligent Systems Laboratory with the University of Guelph, Guelph, ON, Canada. His research interests include robotics, intelligent systems, sensors and multisensor fusion, wireless sensor networks, control systems, machine learning, fuzzy systems, and computational neuroscience.

Dr. Yang has been very active in professional activities. He serves as the Editor-in-Chief for International Journal of Robotics and Automation, and an Associate Editor for IEEE TRANSACTIONS ON CYBERNETICS, IEEE TRANSACTIONS ON ARTIFICIAL INTELLIGENCE, and several other journals. He has been involved in the organization of many international conferences.

Qiaozhi Hua received the M.E. and Ph.D. degrees from the Graduate School of Global Information and Telecommunication Studies, School of Fundamental Science and Engineering, Waseda University, Tokyo, Japan, in 2015 and 2019, respectively.

He is currently a University Lecturer with Computer School of Hubei University of Arts and Science, Xiangyang, China. His research interests include information-centric networking, Internet of Things, artificial intelligence, C-V2X, edge computing, and information security.

Arun Kumar Sangaiha (Member, IEEE) received the M.E. degree in computer science and engineering from Anna University, Chennai, India, in 2007 and the Ph.D. degree in computer science and engineering from Vellore Institute of Technology, Vellore, India, in 2014.

He is currently a Professor and engaged as a supervisor of Ph.D and Master Scholars in the National Yunlin University of Science and Technology, Douliu City, Taiwan. He has visited many research centers and universities in China, Japan, and South Korea for joint collaboration towards research projects and publications. Further, he has authored or coauthored more than 300 SCI journal papers, eight books, one patent, three projects, among one funded by Ministry of IT of India and few international projects (CAS, Guangdong Research fund, Australian Research Council) cost worth of $50 000.

Dr. Sangaiha is the recipient of many awards including the Clarivate Highly Cited Researcher award, Yushan Young Scholar, PIFI-CAS overseas fellowship, Top-10 outstanding researcher, CSI significant Contributor, etc. Also, he is Editor-in-Chief, Book Series Editor, and Associate Editor of various reputed ISI journals.

Tan Guo (Member, IEEE) received the M.S. degree in signal and information processing and the Ph.D. degree in communication and information systems both from Chongqing University (CQU), Chongqing, China, in 2014 and 2017, respectively.

Since 2018, he has been with the School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications (CQUT), Chongqing, China. He is currently a PostDoctoral Fellow with the Macau University of Science and Technology, Taipa, Macao, China. He is recipient of the Macao Young Scholars Program and the Outstanding Chinese and Foreign Youth Exchange Program of China Association of Science and Technology (CAST), Beijing, China. His current research interests include computer vision, pattern recognition, and machine learning.

Keping Yu (Member, IEEE) received the M.E. and Ph.D. degrees in global information and telecommunication studies from the Graduate School of Global Information and Telecommunication Studies, Waseda University, Tokyo, Japan, in 2012 and 2016, respectively.

He was a Research Associate, Junior Researcher, and Researcher with the Global Information and Telecommunication Institute, Waseda University, from 2015 to 2019, 2019 to 2020, and 2020 to 2022, respectively. He is currently an Associate Professor with Hosei University, Tokyo, Japan, and a Visiting Scientist with the RIKEN Center for Advanced Intelligence Project, Saitama, Japan. He has authored or coauthored more than 100 publications including papers in prestigious journal/conferences such as the IEEE Wireless Communications Magazine (WCM), IEEE Communications Magazine (CM), IEEE Network Magazine (NetMag), IEEE Internet of Things Journal (IoTJ), IEEE TRANSACTIONS ON FUZZY SYSTEMS (TFS), IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS (TII), IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS (T-ITS), IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY (TVT), IEEE TRANSACTIONS ON MOBILE COMPUTING (TMC), IEEE Journal of Biomedical and Health Informatics (JBIH), IEEE TRANSACTIONS ON RELIABILITY (TR), IEEE TRANSACTIONS ON NETWORKS AND SERVICE MANAGEMENT (TNSM), IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT (TIM), IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING (TNSE), IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING (GCN), IEEE TRANSACTIONS ON COMPUTATIONAL SOCIAL SYSTEMS (TCSS), IEEE Consumer Electronics Magazine (CEM), IEEE Internet of Things Magazine (IOTM), IEEE International Conference on Communications (ICC), IEEE Global Communications Conference (GLOBECOM), etc. His research interests include smart grids, information-centric networking, the Internet of Things, artificial intelligence, blockchain, and information security.

Dr. Yu has hosted and participated in more than ten projects, is involved in many standardization activities organized by International Telecommunication Union Telecommunication Standardization Sector (ITU-T) and Information-Centric Networking Research Group (ICNRG) of Internet Research Task Force (IRTF), and has contributed to ITU-T Standards Y.3071 and Supplement 35. He is the recipient of the IEEE Outstanding Leadership Award from IEEE BigDataSE 2021, the Best Paper Award from IEEE Consumer Electronics Magazine Award 2022 (1st Place Winner), IEEE International Conference on Frontiers Technology of Information and Computer (ICFTIC) 2021, ITU Kaleidoscope 2020, and the Student Presentation Award from Japan Society for Simulation Technology (JSST) 2014. He is an Associate Editor for IEEE Open Journal of Vehicular Technology, Journal of Intelligent Manufacturing, and Journal of Circuits, Systems and Computers. He has been a Guest Editor for more than 20 journals such as IEEE TCSS. He served as General Cochair and Publicity Cochair of the IEEE Vehicular Technology Conference (VTC2020-Spring) 1st Emerging Blockchain Technology Solutions for Real-World Applications (EBTSRA) workshop, General Cochair of IEEE International Conference on Communications in China (ICCC2020) 2nd EBTSRA workshop, General Cochair of IEEE TrustCom2021 3rd EBTSRA workshop, Session Chair of IEEE ICC2020, ITU Kaleidoscope 2016.