TC-Flow: Chain Flow Scheduling for Advanced Industrial Applications in Time-Sensitive Networks

Dong Yang, Kai Gong, Jie Ren, Weiting Zhang, Wen Wu, and Hongke Zhang

ABSTRACT

Time-sensitive networking (TSN) can help standardize deterministic Ethernet across industrial automation. The deterministic guarantee of TSN is based on network resource scheduling in the unit of flow. However, the state-of-the-art TSN single flow scheduling scheme cannot meet the coordinated scheduling requirements of multiple data flows in advanced industrial applications (e.g., control and safety applications). In this article, we propose a TSN chain flow abstraction, TC-Flow, for a coordinated multiple-flow scheduling model in industrial control and safety applications. Based on the proposed TC-Flow model, we design an offline TC-Flow scheduling algorithm using integer linear programming and an online heuristic TC-Flow scheduling algorithm to handle network dynamics. To deploy the proposed TC-Flow model and scheduling algorithms in the TSN, we design a CF-TSN network architecture that is compatible with the existing TSN single-flow scheduling scheme. Finally, we implement the proposed CF-TSN architecture and TC-Flow scheduling algorithms in real-world network environments. Experimental results show that the proposed scheduling algorithms can increase the number of schedulable flows by 26 percent compared to the state-of-the-art TSN scheduling benchmark.

INTRODUCTION

IEEE 802.1 time-sensitive networking (TSN) is able to provide deterministic communication of time-critical and mission-critical applications over a bridged Ethernet network, which is shared by various kinds of applications having different quality of service (QoS) requirements, that is, time- and/or mission-critical traffic and best effort traffic [1]. Due to the deterministic communication capability, TSN has been considered as one of the representative technologies for the industrial communication network, which is used for data exchange among controllers, sensors, actuators, and other industrial devices [2]. In this scenario, the TSN flow scheduling problem is more than a pure network problem and is often dictated by industrial application-specific semantics.

Unfortunately, the existing TSN literature does not provide complete approaches to express the flow scheduling requirements of advanced industrial applications. Specifically, the abstraction of the current single TSN flow cannot capture the semantics of the coordinated scheduling requirements of multiple flows, where the collective fate of all the flows among a group of industrial devices is more important than that of any individual flow. In essence, the deterministic guarantee of TSN is based on network resource scheduling in the unit of flow. The absence of reasonable abstraction will lead to a lack of global flow scheduling consideration, which results in low network resource utilization.

Actually, the problem of multiple coordinated sub-flows has been studied in other application scenarios. Chowdhury et al. proposed coflow abstraction [3] and coflow scheduling [4] in cluster computing applications in data center networks. Each coflow is a collection of flows between two groups of machines with associated semantics and a collective objective. In the “Network 2030” white paper [5] released by the International Telecommunication Union — Telecommunication Standardization Sector (ITU-T), a new service model, namely coordinated service, is proposed as one of the three basic time-engineered services in future networks. Coordinated service demands tightness of delivery of packets from multiple flows (from the same or multiple sources).

In this article, we focus on industrial scenarios and propose a TSN chain flow (TC-Flow) model, which is a TSN networking abstraction that captures diverse chain flow communication patterns observed in advanced industrial applications. TC-Flow is defined as an end-to-end data flow with delay and jitter requirements, which consists of sequential sub-flows. After all the predecessor sub-flows have completed transmission, the successor sub-flows are triggered to transmit data. According to specific requirements of TC-Flows, each sub-flow can be assigned to a flow type that includes time-triggered (TT) flows, audio video bridging (AVB) flows, and best effort (BE) flows.

The main contributions of this article are as follows. First, we propose a TC-Flow model to abstract chain flow communication patterns in the TSN. Based on the TC-Flow model, a set of novel parameters are designed to accurately describe the coordinated relationship among the sub-flows. Second, to efficiently schedule TC-Flows, we design an offline TC-Flow scheduling algorithm based on integer linear programming (ILP), and
an online heuristic TC-Flow scheduling algorithm to handle network dynamics. Third, we design a chain flow TSN (CF-TSN) architecture to effectively deploy and manage the proposed TC-Flow model and scheduling algorithms. We implement the CF-TSN architecture and TC-Flow scheduling algorithms in real network environments. Experimental results show that the proposed scheduling algorithms increase the number of the schedulable flows by 26 percent compared to the state-of-the-art TSN scheduling benchmark.

The remainder of this article is organized as follows. The TC-Flow model, TC-Flow scheduling algorithms, and CF-TSN network architecture are illustrated in the next three sections. Following that, experimental results are shown, and the article is concluded in the final section.

**TSN Chain Flow**

**Chain Flow in Industrial Applications**

Industrial networks have been designed to satisfy the requirements abstracted from industrial applications. The most important requirements are timeliness and reliability. The International Society of Automation divides industrial applications according to timeliness and reliability into three types: monitoring, control, and safety [6]. The monitoring application is a type of non-timeliness data collection application, in which data flow scheduling is relatively simple. As advanced industrial applications, both control and safety applications involve coordination among multiple devices (e.g., sensors, actuators, and industrial controllers), which require coordinated scheduling for multiple data flows. Figure 1a shows three different flow models of industrial applications: single flow, chain flow, and complicated chain flow. In the first case, one individual flow transmits data from a sensor to an industrial controller, while another flow sends control instructions from the industrial controller to an actuator. Both flows are independent of each other. In the second case, after a sensor sends data to the industrial controller in a periodic or event-triggered manner, the industrial controller will send the analyzed result to an actuator for further operations. In the third case, due to the complexity of advanced industrial applications, multiple sequential sub-flows with semantics complete an industrial application together. Obviously, the second and third cases may increase flow management overhead and flow scheduling complexity. Hence, it is important to develop emerging scheduling algorithms and network architecture that can flexibly support managing and scheduling for TC-Flow to ensure their timeliness and reliability requirements.

In order to make the TC-Flow model in line with industrial characteristics, we refer to IEC/IEEE 60802, which is a TSN standard on industrial automation [7]. A typical TC-Flow with two-level industrial controllers is shown in Fig. 1b. Specifically, the data are uploaded from Machine 3 and Machine 4 to Industrial controller 1, while
Industrial controller 2 drives the actuators in the next production cell to perform production operations in Machine 1 and Machine 2. We define the TC-Flow $f_i$ as a tuple $(s_i, d_i, t_i, DDL_i, REL_i)$, where $p_i$ is the priority of flow $i$, $s_i$ is the source device, $d_i$ is the destination device of flow $i$, $t_i$ is the transmission cycle, and $DDL_i$ is the transmission deadline. To address the coordination problem in TC-Flow, a variable $REL_i$ is defined to represent the coordinated relationship among the sub-flows. The correlation identifier $rel_{f_m, f_a} \in REL_i$ describes connection types of sub-flow $f_m$ and $f_a$ including data collection from sensors and control instruction distribution to actuators, which is shown as follows:

$$rel_{f_m, f_a} = \begin{cases} 
1, & \text{if } f_m \text{ and } f_a \text{ are data collection, or } f_m \text{ and } f_a \text{ are control distribution;} \\
2, & \text{if } f_m \text{ is data collection, and } f_a \text{ is control distribution.}
\end{cases}$$

Based on the correlation identifier $rel_{f_m, f_a}$, the sub-flows in Fig. 1b have clear definition of relationship. For sub-flows 1 and 2, $rel_{f_1, f_2}$ equals 1 because both of them are data collection from Machines 3 and 4. $rel_{f_2, f_1}$ equals 2 means that sub-flow 3 transmits industrial control instruction to the actuators (Machines 1 and 2), which is different from sub-flow 2. For the complicated chain flow scenario in Fig. 1a, the correlation identifier $rel_{f_3, f_a}$ can also be extended to describe the logical connection among sub-flows. For sub-flows from 1 to 4, $rel_{f_{m1}, f_{m2}} \forall m_1, m_2 \in [1, 4], m \neq n$ equals 1 due to the same data collection and transmission relationship. $rel_{f_{m1}, f_{m2}} \forall m_1, m_2 \in [1, 4]$ equals 2 because sub-flow 5 is different from sub-flows from 1 to 4, which transmits control instructions from Industrial controller 2 to an actuator.

The characteristics of the TC-Flow $f_i$ are summarized as follows:

- One TC-Flow scheduling $S_f$ is the coordinat-ed scheduling for multiple sub-flows $\cup_{i \in S_f} S_i$
- The sub-flow has a strict logical relationship $REL$ that is determined by TC-Flows
- The scheduling procedure needs to meet the requirements of both TC-Flows and sub-flows.

**Flow Scheduling in TSN**

To deploy flows that satisfy deterministic requirements in TSN, an effective scheduling algorithm is necessary. Existing TSN scheduling algorithms are mainly divided into two categories: solver-based scheduling and heuristic scheduling algorithms. For solver-based scheduling algorithms, the widely adopted optimization solvers are satisfiability modulo theories (SMT) and ILP. Craciunas et al. [8] investigated the computation problem of fully deterministic scheduling for a single flow, and analyzed key functional parameters for 802.1Qbv (i.e., TSN standard enhancement for scheduled traffic). Nayak et al. [9] proposed algorithms for incrementally adding a time-triggered single flow in time-sensitive software-defined networking (SDN). Z. Li et al. [10] proposed an enhanced reconfiguration scheduling mechanism based SMT to reduce duration time of updates. In [11], a no-wait job shop scheduling scheme was used to obtain the scheduling scheme in IEEE 802.1Qbv. An important feature of all these methods is that the flow routing problem is separated from flow scheduling, and flow routing is required before the calculation of flow scheduling. For heuristic-algorithm-based scheduling, Schweissguth et al. [12] studied the joint routing and scheduling problem and proposed an ILP-based solution and a list scheduling-based heuristic algorithm. In [13], the heuristic-algorithm-based scheduling scheme was developed to balance the transmission and network update. In [14, 15], an artificial intelligence (AI)-based network resource scheduling scheme is a possible solution in TSN.
Although the above scheduling schemes effectively improve the scheduling capacity for the single-flow case, they cannot be directly applied to TC-Flow scheduling due to the lack of TC-Flow global feature analysis. Therefore, we propose two efficient scheduling algorithms to schedule TC-Flows in the following section.

SCHEDULING FOR TC-FLOWS

To optimize network resource allocation for the TC-Flow model, we discuss TC-Flow scheduling constraints and propose two scheduling algorithms, an ILP-based offline scheduling algorithm and a heuristic online scheduling algorithm, which are described below.

ILP-BASED OFFLINE SCHEDULING ALGORITHM

Based on the feature of TC-Flows (including link connection, delay, jitter, etc.), we propose an ILP-based offline scheduling algorithm to meet the requirements of TC-Flows, which includes three steps: network constraint initialization, routing generation, and scheduling calculation.

Network Constraint Initialization: Network constraint initialization consists of two parts: TC-Flow analysis and constraints generation. For TC-Flow $F_i$, the offline algorithm first analyzes TC-Flow’s information and calculates the period of cycle slot, which is the least common multiple of all sub-flow periods. Based on the information and requirements of TC-Flows, a set of constraints are designed including frame constraints, connection constraints, latency constraints, resource constraints and so on. Specifically, the constraints mainly describe the basic information of TC-Flow $F_i$ including priority, cycle, frame length, and so on. The connection constraints are proposed to express the strict transmission sequence of the sub-flows based on the correlation identifier $rel_{i,m_n}$, $i \in REL$ and $R_i[|v_a,v_b|]$, which is a link identifier showing the status of path $[v_a, v_b]$ when scheduling TC-Flow $F_i$. To avoid transmission conflict among sub-flows $F_{i(v_a,v_b)}$ and $F_{i(v_b,v_a)}$, the connection constraints are described as follows:

$$
\forall F_{i(v_a,v_b)}: \quad 0 < \sum_{v_b \in (v_a,v_b)} rel_{i(m_n,v_b)} \times R_{i(m_n)}[|v_b,v_a|] \\
\leq \sum_{v_b \in (v_a,v_b)} rel_{i(m_n,v_b)} \times R_{i(m_n)}[|v_b,v_a|] \\
\leq \Delta \times \sum_{v_b \in (v_a,v_b)} rel_{i(m_n,v_b)} \times R_{i(m_n)}[|v_b,v_a|] 
$$

(2)

Here, TC-Flow $F_{i(v_a,v_b)}$ sends data from $v_a$ to $v_b$, and $v_i$ represents the industrial controller that analyzes data from sensors ($i(v_a,v_b)$) and sends control instructions to actuators ($i(v_b,v_a)$). Equation 2 shows that at least one sub-flow should be input into the industrial controller, and the sub-flow is transmitted to the actuator after all sensors’ data in the controller are analyzed. $\Delta$ is a sufficiently large integer commonly used to formulate the conditional constraints in ILP. To ensure transmission order of sub-flows in Fig. 1b, the sum of $rel \times R_{i(v_a)}$ for $F_i$ is less than that for $F_j$.

The latency constraints are time requirements of TC-Flows and sub-flows, which are all considered as input during the ILP calculation phase. The resource constraints mainly include network bandwidth and link status prepared for routing decisions. These constraints describe the main features of TC-Flow, which are the main inputs of the scheduling calculation.

Routing Generation: The sub-flows of TC-Flow bring large decision space in the routing generation phase. To reduce the solution space, the network topology needs to be preprocessed so that the unrelated switches and links are removed (i.e., the ones through which TC-Flows do not transmit). After topology pre-processing, the computational complexity of routing generation is reduced, which further simplifies TC-Flow scheduling calculation.

Scheduling Calculation: Based on the above network constraints and routing results, solvers like ILP are used to calculate the scheduling result. The objective function of scheduling calculation in ILP is to optimize the number of scheduled flows while minimizing the link bandwidth occupancy rate. The calculation process can be executed multiple times to obtain the best scheduling result.

HEURISTIC ONLINE SCHEDULING ALGORITHM

To handle network dynamics, we propose a heuristic online scheduling algorithm using the offline scheduling results as its initial reference value. As shown in Fig. 2, the heuristic algorithm contains two parts: neighborhood search and tabu list.

Neighborhood Search: For the neighborhood search phase, the online algorithm is mainly to generate a neighborhood solution space and optimal solutions. Specifically, based on flow information and routing results, we first initialize the initial scheduling, which is current scheduling denoted by $s0$ and the best scheduling denoted by $bs0$. Initialization status of the tabu list is void.

Then we generate the set of $s0$ neighbor scheduling $S$. To minimize the end-to-end delay, the best scheduling $bs1$ is solved from $S$. $bs1$ is subject to both the requirements of TC-Flows and the related connection constraints among sub-flows.

Tabu List: In subsequent iterations, we choose the time slot scheduling that conforms to the tabu list or meets the required standard in the neighborhood with the shortest processing delay. The tabu list contains a list of flows that are identified as the best scheduled flows in the previous iteration. When scheduling TC-Flow with different sub-flows, the tabu lists are shared to reduce local cycle calculation.

If the best scheduling $bs1$ exists in the tabu list, the algorithm will make a decision that $bs1$ remains in the list. If $bs1$ does not remain in the list, the next best solution $ns1$ is selected from the candidate set. If $bs1$ is not in the tabu list, it will be compared with the current best scheduling, and the best one will be selected. Because the tabu search effectively avoids the problem of a local optimal solution through the tabu list, the proposed heuristic algorithm can quickly obtain feasible online scheduling results in TSN.
**CF-TSN Architecture for TC-Flows**

To effectively deploy and manage the proposed TC-Flow model and scheduling algorithms, we propose a CF-TSN architecture as shown in Fig. 3. Leveraging the SDN technique, the proposed architecture is divided into a control plane and a data plane, which enables users to flexibly operate various industrial applications and reduces the complexity of configuration. The control plane is an advanced “brain” composed of high-performance network computing servers. Two key functions of the control plane are TC-Flow intelligent analysis in the CF-TSN pre-processing phase and scheduling calculation in the CF-TSN offline scheduling phase, which correspond to an intelligent flow analyzer and an SDN-based central network controller (SD-CNC).

**CF-TSN Control Plane**

To intelligently analyze TC-Flows, multi-dimensional flow parsing is proposed to abstract the deterministic requirements of TC-Flow services and analyze flow basis information. Specifically, the deterministic requirements include delay, jitter, loss rate, and so on. The flow information model includes the flow size, transmission rate, source and destination, and deterministic coordination of TC-Flow and its sub-flows. Based on these information and requirements, a set of network constraints are generated as inputs to the SD-CNC. The above information is stored in a CF-TSN database which can be used in subsequent procedures. Based on multi-dimensional flow parsing, the TSN traffic category is mapped for each TC-Flow to grant the service level agreement. The TSN traffic class assigned to each TC-Flow is encoded in the immutable field of the packet header.

In addition, based on the features of TC-Flows, network topology is optimized to reduce the complexity of routing calculation. The devices that are unrelated to the flows are not considered according to diverse scenarios. In the routing calculation, the industrial controller is considered as a specific device with the data analysis function, which can not only collect the data of industrial devices but also deploy the analysis strategy of applications. Based on the pre-processing phase, the routing results are calculated using routing algorithms. After the analysis of TC-Flows, routing results are sent to CF-CNC with other scheduling constraints.

The SD-CNC acts as the global network controller in the CF-TSN, which controls network devices and collects telemetry data about the link status. Furthermore, the SD-CNC is responsible for calculating feasible routing results and time slot scheduling to meet the TC-Flow requirements. Based on the routing results, SD-CNC uses ILP to generate TC-Flow scheduling results, that is, the gate control list (GCL). According to the priorities of different applications, the network controller arranges different service queues flexibly based on...
the GCL. Configurations calculated by SD-CNC are transmitted to TSN switches. With the larger network scale, the TSN topology would be divided into multiple domains. By using an open network operating system, the SD-CNC is designed as a multi-level control framework for distributive network resource management among multiple domains, which improves the feasibility and scalability of CF-TSN.

**Procedure of TC-Flow Scheduling**

As shown in Fig. 4, to efficiently deploy the proposed scheduling algorithms in the CF-TSN, we divide the procedure of TC-Flow scheduling into three phases: CF-TSN network initialization phase, configuration phase, and operation phase.

**Network Initialization Phase:** The network initialization phase includes registrations of network devices, and collection of TC-Flow information and link status. First, all devices in the network should be registered to the CF-CUC. Then a clock synchronization function is used to calibrate the clock among devices. Meanwhile, the control session is established to build up the bridge among the network controller, switches, and devices (Steps 1–5). For topology discovery and link status collection, the SD-CNC periodically sends detection packets to each network device, which feeds back path information to the network controller. Based on the path information, the network controller can form a global network topology (Step 6). In addition, the link status is updated via link layer discovery protocol (LLDP) and stored in a database (Step 7).

**Network Configuration Phase:** The network configuration phase consists of the generation and deployment of schedule results. First, a
The real-world network environment for CF-TSN.

TC-Flow $F_i$, denoted by $<P_i, S_i, D_i, T_i, DDL_i, REL_i>$, initiates a flow requisition from host devices to CF-CUC (Step 8). The CF-CUC collects the TC-Flow requirements (priority $P_i$, flow correlation $REL_i$, available bandwidth, and delay requirement $DDL_i$) and sends them to the control plane, where flow constraints are analyzed for TC-Flow $F_i$ (Steps 9–10). Second, the network topology is built up by LLDP and then optimized to generate the routing result based on routing algorithms. The TC-Flow information with flow constraints and routing result is regarded as the input of SD-CNC, which calculates the starting time of data packet transmission to generate the GCL. (Steps 11–15).

Network Operation Phase: In this phase, data flows are transmitted in the network. According to ingress policing protocol (802.1Qci), data packets inconsistent with the QoS requirements are discarded. Then switches forward the packets to the destination hop by hop. Moreover, the TSN shaping and scheduling strategies (802.1Qav, Qbv, Qch, etc.) are set at the TSN switch egress ports, such as time-aware shaper and queue discipline of bandwidth limitation.

CASE STUDY

EXPERIMENTAL SETUP

In this section, we provide a case study for TC-Flow scheduling, aiming to evaluate the schedulability of the proposed algorithms. As shown in Fig. 5, the proposed architecture is validated in a real TSN environment that consists of TSN switches, SD-CNC, and end devices used as industrial devices. Each TSN switch is equipped with an Intel Core i7-6700 CPU@3.40 GHz and 1.6 GB of RAM and Intel Ethernet Server Adapter I210, which supports TSN functions configured by Linux Traffic Control (TC) tools. In addition, we use the ILP-Solver to obtain network scheduling decisions, and the model of ILP-Solver is executed in the SD-CNC, which is a server equipped with an Intel Core i5-6500 quad-core 3.20 GHz CPU and 8 GB of RAM.

EXECUTION TIME OF ALGORITHMS

We set up the number of TC-Flows from 45 to 180. The network scale is no more than 7 hops, and in each TC-Flow no more than 3 sub-flows can be separated. The number of iterations in the ILP-Solver is set to be 20. In addition, for the heuristic algorithm, the number of iterations is set to no more than 20 to obtain better online scheduling results compared to offline scheduling. Moreover, we set up an experiment to compare the proposed scheduling algorithms with the traditional SMT-based algorithm in which the two sub-flows of a TC-Flow are scheduled separately.

The execution time of the TC-Flow scheduling algorithm is shown in Fig. 6a. When the number of TC-Flows increases, the execution time of the scheduling algorithm increases in a polynomial manner. As shown on the right Y-axis of Fig. 6a, the normalized gain of algorithm execution time indicates the ratio between the proposed algorithm and the compared algorithm in terms of execution time.

Specifically, when we schedule 90 TC-Flows, the execution time of the benchmark algorithm is 8 percent higher than that of the proposed ILP offline scheduling algorithm and 30 percent higher than that of the proposed online scheduling algorithm. The results demonstrate that the proposed algorithms can effectively reduce the execution time of TC-Flows. When the number of TC-Flows is more than 120, the maximal scheduling capacities of all algorithms have been reached, such that the execution time of all algorithms are prohibitively high and close to each other.

SCHEDULABILITY

The schedulability metric is defined as the ratio between the number of successfully scheduled flows and the number of total flows, which represents the scheduling capacity of the proposed algorithms. We increase the number of flows in the scheduling phase until the network fails to schedule TC-Flows. As shown in Fig. 6b, when the number of TC-Flows increases to 120 flows, the schedulability of the non-CF-TSN algorithm decreases to 0. For the ILP-based offline algorithm, the schedulability decreases to 0 when the successfully scheduled TC-Flows are 135, an increase of 13 percent compared to the non-CF-TSN. In addition, the schedulability of the proposed online scheduling algorithm decreases to 0 when the successfully scheduled TC-Flows are 152, an increase of 26 percent compared to the non-CF-TSN scheduling algorithm.

END-TO-END DELAY AND JITTER

We set up the number of TC-Flows from 15 to 150 for scheduling in the CF-TSN because the schedulability of algorithms decreases quickly when there are more than 150 TC-Flows. To guarantee the transmission delay and jitter performance, the scheduled TC-Flows are set as the same number of hops (no more than 7 hops). For the end-to-end delay, we obtain the raw data and calculate the worst case value for performance evaluation. Then the jitter is defined as the range of end-to-end delay.

As shown in Fig. 6c, the proposed offline and online scheduling algorithms achieve average 800 ms of end-to-end delay performance, and the jitters of the proposed algorithms are roughly 120 ms, which conforms to the time requirements of TSN. Therefore, the proposed algorithms can achieve good schedulability performance while guaranteeing timeliness and reliability of TSN.
CONCLUSION
In this article, we have studied the coordinated scheduling problem for chain flows in TSN, named TC-Flow. To the best of our knowledge, this article is the first work focused on chain flow scheduling for advanced industrial applications. Based on the feature of TC-Flow, the offline and online scheduling algorithms have been proposed to improve the scheduling capacity. To deploy the TC-Flows and algorithms in TSN and ensure their deterministic communication, a novel CF-TSN network architecture has been proposed. Finally, we have evaluated the performance of the proposed model, algorithm, and architecture in real network environments, and the experimental results have validated their effectiveness.

ACKNOWLEDGMENTS
The work is supported in part by the National Key Research and Development Program of China under Grant 2018YFB1702000, in part by the Fundamental Research Funds for the Central Universities under Grant 2020YJS017, and in part by the Major Key Project of PCL under Grant PCL2022YD4.

REFERENCES

BIOGRAPHIES
Dong Yang [M’11] (dyang@bjtu.edu.cn) received his Ph.D. degree in communications and information science from Beijing Jiaotong University, China, in 2009. From March 2009 to June 2010, he was a postdoctoral research associate at Jonkop-
Kai Gong (18111013@bjtu.edu.cn) is currently pursuing a Ph.D. degree in communication and information systems at the National Engineering Laboratory for Next Generation Internet Technology in China. His research interests include time-sensitive networks, artificial intelligence, and the Internet of Things.

Jie Ren (21111028@bjtu.edu.cn) is currently pursuing a Ph.D. degree in communication and information systems at the National Engineering Laboratory for Next Generation Internet Technology in China. His research interests include time-sensitive networks and the Internet of Things.

Weiting Zhang [S’20] (wtzhang@bjtu.edu.cn) received his Ph.D. degree in communication and information systems from Beijing Jiaotong University in 2021. From November 2019 to November 2020, he was a visiting Ph.D. student with the Department of Electrical and Computer Engineering, University of Waterloo, Ontario, Canada. Starting in December 2021, he is an associate professor with the School of Electronic and Information Engineering, Beijing Jiaotong University. His research interests include the Industrial Internet of Things, mobile edge computing, and machine learning for wireless networks.

Wen Wu [S’13, M’20] (wwu02@pcl.ac.cn) received his Ph.D. degree in electrical and computer engineering from University of Waterloo in 2019. He received his B.E. degree in information engineering from South China University of Technology, Guangzhou, and his M.E. degree in electrical engineering from the University of Science and Technology of China, Hefei, in 2012 and 2015, respectively. He worked as a postdoctoral fellow with the Department of Electrical and Computer Engineering, University of Waterloo. He is currently an associate researcher at the Frontier Research Center, Peng Cheng Laboratory, Shenzhen, China. His research interests include 6G networks, pervasive network intelligence, digital twin, and network virtualization.

Hongke Zhang [M’13, SM’16, F’20] (hkzhang@bjtu.edu.cn) is a full professor with the School of Electronic and Information Engineering, Beijing Jiaotong University. He directs the National Engineering Laboratory for Next Generation Internet Technology in China. His research has resulted in many papers, books, patents, systems, and equipment in the areas of communications and computer networks. He is the author of more than 10 books and the holder of more than 70 patents. He is the Chief Scientist of the National Basic Research Program of China (973 Program). He serves as an Associate Editor for the IEEE Internet of Things Journal and IEEE Transactions on Network and Service Management.