Adaptive Group Routing and Scheduling in Multicast Time-Sensitive Networks

QINGHAN YU\textsuperscript{D}, (Student Member, IEEE), AND MING GU

Key Laboratory for Information System Security (KLISS), Ministry of Education, Tsinghua University, Beijing 100084, China
Beijing National Research Center for Information Science and Technology (BNRist), Tsinghua University, Beijing 100084, China
School of Software, Tsinghua University, Beijing 100084, China

Corresponding author: Qinghan Yu (yuqinghan10@gmail.com)

This research was sponsored in part by the National Natural Science Foundation of China (NSFC) Program (No. 61527812), the National Science and Technology Major Project of China (No. 2016ZX01038101), the Ministry of Industry and Information Technology (MIIT) Information Technology (IT) Funds of China (Research and Application of TCN Key Technologies), and the National Key R&D Program of China (No. 2016QY07X1402).

ABSTRACT Traditionally, industrial networks rely on field bus to support real-time applications. In the trend of industrial internet of things (IIoT), traffic load has increased dramatically, and industrial networks are moving towards time-sensitive networking (TSN), which is based on the widely used Ethernet. In TSN, the time-triggered mechanism is a promising technology for real-time control due to its ability to provide deterministic latency and jitter guarantees. However, this mechanism is mainly based on offline computed schedules, making it difficult to satisfy dynamic requirements in data transmission of IIoT. In this paper, we propose an adaptive group routing and scheduling (AGRS) approach in multicast time-sensitive networks. The framework of this approach contains three phases including pre-processing, schedule synthesis, and post-processing phases. While the pre-processing phase first prunes unnecessary links and adaptively groups qualified complete subgraphs of the network to obtain a simplified topology, the schedule synthesis phase then jointly considers routing and scheduling processes based on the simplified topology to improve schedulability. Finally, the post-processing phase handles the scheduling results of previous phases to generate a feasible schedule. Experiments show that our approach improves schedulability by 68\% and reduces execution time by 74\% for multicast time-sensitive networks compared with existing approaches.

INDEX TERMS Integer linear programming, real-time scheduling, time-sensitive networks.

I. INTRODUCTION

With the proliferation of distributed sensors, surveillance cameras and actuators, cyber-physical systems (CPS) emerge as the mechanisms monitoring and controlling physical processes, which transmit data and commands through computer networks. Typically, CPS have stringent timing requirements, and the performance of networks has a significant impact on the quality of CPS control. For instance, a motion control application in industrial automation may require that the time of control cycles is measured in a few microseconds [1]. In addition, a majority of industrial applications, such as trains and ships, use multicast networks for system control. Therefore, multicast time-sensitive networks that provide deterministic latency and latency variance (jitter) are urgently desired in order to ensure real-time control for CPS.

Generally, there are two types of application data in industrial networks: real-time data for system control and best-effort data for ancillary service. The real-time data, with fixed period and size, must arrive at all the destinations within its deadline and satisfy the jitter constraints. In contrast, the best-effort data requires no timing guarantees. Traditionally, industry networks achieve deterministic guarantees for real-time data by dedicated field bus [2]. However, traffic load has increased dramatically in the trend of industrial internet of things (IIoT). Therefore, the low bandwidth, typically 1Mbps, and limited maximum number of supported devices number in the obsolete field bus cannot address this new tendency.

In the last decade, Ethernet standardized in IEEE 802.3 [3] has been widely used in the local area networks (LAN) for offices and families. Compared to the networks based on field bus, Ethernet brings higher bandwidth up to 100 Gbps and better compatibility with existing network devices. Although the IEEE 802.3 Ethernet is initially designed for best-effort
data and thus cannot provide real-time communication services [4], various real-time Ethernet technologies, such as Profinet [5], EtherCAT [6], FTT [7], TT-Ethernet [8], have been proposed to achieve hard real-time guarantees for Ethernet. In addition, the Deterministic Networking (DetNet) Working group and IEEE 802.1 Time-Sensitive Networking (TSN) Task Group strive to promote the standardization of real-time networks to provide deterministic end-to-end latency and jitter for time-sensitive applications. Moreover, the recently released IEEE 802.1 Qbv [9] standard schedules real-time data in a time-triggered mechanism. The time-triggered mechanism provides latency and jitter guarantees by following two steps: (1) Before network deployment, a schedule synthesis procedure receives the timing parameters of real-time flows as input and generates a valid scheduling table for these flows. (2) While the network is operating, switches schedule the real-time flows according to the pre-computed scheduling table. As the offline scheduling table is validated to satisfy the real-time constraints for all the real-time flows during the synthesis stage, the switches perform deterministic communication according to the table, thus providing latency and jitter guarantees.

Although time-triggered communication is well-established [10]–[12], the trend of IIoT brings new challenges from dynamic applications [13]. These dynamic applications occasionally introduce new requirements of data transmission in time-sensitive networks, for instance, adding a plug-and-play device in the production line [14]. Therefore, a novel scheduling approach is required to schedule the new requirements without disturbing existing schedule, as schedule modifications may lead to a violation of latency and jitter bounds for the existing flows, resulting in unpredictable consequences [15]. Moreover, this scheduling approach is able to take full advantage of available bandwidth to accept new transfer requests as much as possible, and achieve low execution time to provide a rapid response to these new requests.

In this paper, an adaptive group routing and scheduling approach is explored to maintain the quality of service (QoS) of dynamic applications in multicast time-sensitive networks. In this approach, the network model and the routing and scheduling problem are first formalized respectively. As computational complexity is closely related to the topology complexity, a pre-processing phase is devised to simplify the network topology, which prunes unnecessary links and adaptively groups qualified complete subgraphs. While existing approaches [10], [11], [16] use the Steiner Tree as default routes, a routing and scheduling (RS) procedure is devised in the schedule synthesis phase, which jointly considers routing and scheduling processes to improve schedulability. The RS procedure is based on the simplified topology processed by the former phase, and thus the execution time of schedule synthesis is sharply reduced. Furthermore, a post-processing phase is proposed to handle the results of the schedule synthesis phase, which uses a counter example strategy to generate a feasible schedule. Finally, the experiments show that our approach can achieve better schedulability and lower execution time for multicast time-sensitive networks compared with existing approaches. In summary, the main contributions as follows:

- An adaptive link grouping strategy is proposed to reduce the execution time of the schedule synthesis process.
- A routing and scheduling (RS) procedure that jointly considers routing and scheduling is developed to improve the schedulability for multicast time-sensitive networks.
- A 24-port TSN switch supporting on-the-fly schedule reconfiguration is implemented, and the approach is validated on the hardware platform.

The paper is organized as follows: In Section II, the related work is discussed. Then, the network model is illustrated and the routing and scheduling problem is formalized for the multicast time-sensitive networks in Section III. After the AGRS framework is discussed in Section IV, the pre-processing phase, schedule synthesis phase and post-processing phase are presented respectively in Section V, VI, and VII. Finally, experimental results show the performance in Section VIII, and the paper is concluded in Section IX.

II. RELATED WORK

Message scheduling in real-time networks is known to be an NP-complete problem [12], and various researchers have explored methods of improving and optimizing schedule synthesis for decades. Traditionally, real-time networks are analyzed in regards to worst-case end-to-end delay through network calculus [17], [18]. The major drawback of this approach is that the delay analysis is based on the pre-assigned priorities and the arrival patterns of competing flows, which makes it difficult for the compositional system design and the communication isolation.

Due to its deterministic and composability [19], [20], time-triggered networks, such as FTT-CAN [21] and FlexRay [22], have been investigated in detail. However, these approaches are based on the traditional field bus, and thus are not suitable for large-scale and high-bandwidth networks based on the widely used Ethernet. For multihop time-sensitive networks based on the Ethernet, the approaches, such as TT-Ethernet [10] and Profinet [12], have been proposed to synthesize offline schedules. These approaches first generate a sufficient constraint set of transfer requirements as the input, and then use satisfiability modulo theory (SMT) or integer linear programming (ILP) to synthesize a schedule satisfying these constraints. For large-scale networks, these approaches have been extended to improve capability and performance [23], [24]. Furthermore, these approaches have also been extended to generate gate control list for IEEE 802.1 Qbv networks [16], [25]. Heuristic approaches [12], [26] to scheduling the flows based on the constraint set have been discussed. F. Dürre, et al. [27] formalize the schedule synthesis problem as a no-wait packet
scheduling problem (NWPS), and use Tabu search to generate near-optimal solutions.

All of the aforementioned approaches separate routing and scheduling processes, and assume that the routes of flows are given as a prior. Generally, the Steiner Tree is used as the default routes. However, the separation of routing and scheduling processes reduces the solution space, and critical paths may become the bottlenecks of scheduling. Schweissguth et al. [28] and Smirnov et al. [29] propose approaches that jointly consider routing and scheduling for unicast time-triggered networks. However, due to lack of optimization, the greatly increased solution space makes it unsuitable for large-scale networks. Nayak et al. [30] propose an approach that schedules the transmission of time-triggered flows only on the hosts. Although online scheduling for dynamic applications can be achieved by this approach, the network diameter of this approach is restricted to seven hops, which also fails to satisfy the requirements of large-scale networks.

As a contrast, our approach not only jointly considers routing and scheduling processes to improve schedulability, but also optimizes the solving process to reduce the execution time for large-scale networks.

III. SYSTEM MODEL AND PROBLEM DEFINITION

In this section, the system model of multicast time-sensitive networks are first formalized, and then the routing and scheduling problems considered in this paper are defined.

A. SYSTEM MODEL

The time-sensitive network is modeled as an unweighted directed graph \( G = (V, E) \), where \( V \) is the set of network nodes, consisting of the set of switches \( SW \) and the set of end devices \( ED \), and \( E = \{ (v_i, v_j) \mid v_i, v_j \in V, \text{ and } v_i, v_j \text{ are connected by a link} \} \) is the set of directional network links. \( |V| \) and \( |E| \) are the number of nodes and links respectively. A link \( (v_i, v_j) \) of \( E \) transmits flows from the source \( v_i \) to the destination \( v_j \), and \( B_{(v_i, v_j)} \) is the bandwidth utilization of the link \( (v_i, v_j) \). The whole network is assumed to be precisely synchronized.

A set of multicast flows \( F \) with latency and jitter constraints is considered in the time sensitive networks. Each flow \( f_a \in F \) is denoted by a tuple \( < s_a, D_a, p_a, l_a, ddl_a > \), where \( s_a \in V \) is the source, \( D_a \subseteq V \) is the set of destinations, \( p_a \) is the transmission period, \( l_a \) is the maximum length of the flow and \( ddl_a \) is the transmission deadline. Without loss of generality, all these parameters are assumed as positive integers. To ensure the real-time features of the networks, the flow \( f_a \) is required to transmit from the source \( s_a \) to all the destinations \( D_a \) within its deadline \( ddl_a \).

B. PROBLEM DEFINITION

Two types of variables are introduced to illustrate the routing and scheduling problems respectively. To address the routing problems, a binary routing variable \( lo_{(v_i, v_j), f_a} \) is introduced to denote whether the link \( (v_i, v_j) \in E \) is occupied by the flow \( f_a \in F \):

\[
lo_{(v_i, v_j), f_a} = \begin{cases} 1 & \text{flow } f_a \text{ is routed via } (v_i, v_j) \\ 0 & \text{flow } f_a \text{ is not routed via } (v_i, v_j) \end{cases}
\]

To address the scheduling problems, an integer scheduling variable \( \phi_{(v_i, v_j), f_a} \) is introduced with the domain \([0, p_a]\) to denote the scheduling results. If the flow \( f_a \) is routed via the link \( (v_i, v_j) \), \( lo_{(v_i, v_j), f_a} = 1 \), the scheduling variable \( \phi_{(v_i, v_j), f_a} \in [1, p_a] \) is the time point to send the flow \( f_a \) on the link \( (v_i, v_j) \). Otherwise, \( \phi_{(v_i, v_j), f_a} = 0 \). In addition, a schedule \( S \) is uniquely determined by the tuples below:

\[
S = \{ < lo_{(v_i, v_j), f_a}, \phi_{(v_i, v_j), f_a} > \mid \forall (v_i, v_j) \in E, \forall f_a \in F \}
\]

Due to transmission collision and lack of latency planning, an arbitrary schedule cannot provide real-time guarantees. To ensure this feature, the tuples of a schedule \( S \) should satisfy certain constraints, which are called time-sensitive constraints. The routing and scheduling problems in multicast time-sensitive networks are defined as synthesizing a valid schedule \( S_* \), whose tuples satisfy all the time-sensitive constraints. The time-sensitive constraints are discussed in detail in Section VI.

IV. AGRS FRAMEWORK

As described above, the aim of routing and scheduling is to generate a valid schedule whose tuples satisfy all the time-sensitive constraints. In this section, we illustrate the framework of the adaptive group routing and scheduling (AGRS) approach to generate a valid schedule. As shown in Fig. 1, the framework contains the pre-processing, schedule synthesis and post-processing phases, which are described in detail below.

1) PRE-PROCESSING PHASE

As computational complexity is closely related to the complexity of networks, the pre-processing phase simplifies the network topologies to reduce computational complexity for future scheduling process. This phase contains two steps including the pruning step and link grouping step. While the pruning step removes the unnecessary links that flows can never be transmitted through, the link grouping step merges the qualified complete subgraphs of the network to further simplify the topology.

2) SCHEDULE SYNTHESIS PHASE

The schedule synthesis phase jointly considers the routing and scheduling problems. That is, the routes of the flows in this phase are not determined as a prior, but are dynamically generated during the scheduling procedure. Therefore, the solution space is sharply enlarged. If a valid schedule exists, the approach is able to find the schedule and its corresponding routes, which may be pruned by the approaches with fixed routes. Through this strategy, the AGRS approach takes full advantages of available bandwidth, and improves schedulability. Furthermore, this phase synthesizes schedules based on the simplified topologies generated by the
3) POST-PROCESSING PHASE
The post-processing phase handles the results of the schedule synthesis phase. This phase contains two steps including the splitting step and the schedule mapping step. If a valid schedule of the simplified topology is able to be synthesized, the schedule mapping step converts the valid schedule to an equivalent schedule of the original topology. Otherwise, the splitting step refines the simplified topology for future scheduling, during which the link failing to satisfy the constraints is split out of the group.

It should be noted that the schedule synthesis phase itself is a fully featured scheduling approach, taking the original topology as input. However, this phase considers all the possible combinations of routing and scheduling variables, and thus enlarges the problem space dramatically. In the meantime, the number of constraints increases in a quadratic order with respect to the fixed routing methods, which makes it unsuitable for large-scale networks. Furthermore, the large problem space and huge number of constraints sharply increase the execution time of schedule synthesis procedure, resulting in a failure to provide a fast response for the transfer requests of dynamic applications. Therefore, the pre-processing and post-processing phases are introduced to prune and merge the network links without affecting the solution space. On the other hand, the reduced number of links decreases the constraint number of routing and scheduling variables enormously, and thus accelerates the schedule synthesis for the dynamic applications. On the other hand, the maintained solution space preserves the schedulability. Therefore, the AGRS approach can achieve better schedulability and lower execution time compared with existing approaches.

V. PRE-PROCESSING PHASE
In this section, the two steps, pruning and link grouping, in the pre-processing phase are illustrated, which simplify the network topologies for future scheduling process.

A. PRUNING
While the routing and scheduling variables of a flow are allocated for each link in the system model, some of them may be unnecessary in the schedule synthesis procedure. For example, there is always one routing variable $l_{[v_i, s_a]}$, $f_a$ for the link $[v_i, s_a]$ where $s_a$ is the source of the flow $f_a$ and $[v_i, s_a] \in E$, even though the flow $f_a$ can never be transmitted via the link $[v_i, s_a]$. In the pruning step of the pre-processing phase, all the links that can never be part of the routes are pruned, as well as their corresponding routing and scheduling variables. The links connected to the three types of nodes shown below should be removed:

1) SOURCE
The incoming links to the source $s_a$ of a flow $f_a$ should be pruned. Moreover, the predecessor link of the pruned one should also be removed iteratively, if the link has only two predecessors.

2) DESTINATION
The outgoing links from all the destinations $D_a$ of a flow $f_a$ should be pruned. Moreover, the successor link of the pruned one should also be removed iteratively, if the link has only two successors.

3) OTHER END DEVICES
The links that connect to the other end devices should be pruned. Moreover, the adjacent link should also be removed iteratively, if the link has only two successors.

After pruning the unnecessary links described above, the system model can be simplified by about 20%, which has a quadratic impact on the constraint number. Furthermore, the following phases are based on the simplified topology instead of the original one.

B. LINK GROUPING
Although the unnecessary links are removed in the pruning step, the topology is still complicated for jointly considering
the routing and scheduling processes. In order to further reduce the amount of all possible combinations of routing and scheduling, we adaptively group the qualified complete subgraphs and merge the relative links in the link grouping step. A bandwidth threshold $B_T$ is introduced to avoid over-merging, and only links with bandwidth utilization below the threshold are merged. The pseudo-code of the procedure is given as follows.

**Procedure 1 Link Grouping Procedure**

**Input:** Network $G$, bandwidth threshold $B_T$

**Output:** Grouped Network $G'$

1. $X \leftarrow \emptyset$
2. $G' \leftarrow$ remove the links in $G$ with the bandwidth $\geq B_T$
3. Compute the complete subgraph set $R$ of $G'$
4. While $R \neq \emptyset$ do
   5. Compute the subset $R_m \subseteq R$ with the most edges
   6. While $R_m \neq \emptyset$ do
      7. Choose $r \in R_m$ with the least average bandwidth
      8. If $r \cap X = \emptyset$ then
         9. Merge the links in $r$ into a group
         10. Add the links of $r$ into $X$
      end if
   end while
12. end while
14. Merge the links between the link groups
15. return $G'$

The link grouping procedure merges the qualified complete subgraphs within the bandwidth threshold $B_T$ into link groups, where the link set $X$ stores the already merged links. The links that fail to satisfy the bandwidth requirement are first removed (line 2), and the famous Bron-Kerbosch algorithm [31] is used to compute the complete subgraph set $R$ (line 3). It should be noted that the complete subgraph set of $G$ can be computed offline and filtered out the undesirable subgraphs online to get $R$ in order to reduce the calculation time. Then, the complete subgraphs with the most edges and least average bandwidth containing no merged links are iteratively grouped into link groups (line 6-13). Finally, the links between the link groups are merged (line 14), and a further simplified network $G'$ is generated, which is used as input for the schedule synthesis phase. As the maximum number of complete subgraphs is $((|V| - d) \cdot \frac{3^5}{2}$ where $d$ is degeneracy of the graph $G$, the time complexity of the link grouping procedure is $O((|V| - d) \cdot \frac{3^5}{2} \cdot |E|)$.

**VI. SCHEDULING SYNTHESIS PHASE**

In this section, the routing and scheduling (RS) procedure in the schedule synthesis phase are illustrated to generate a valid schedule, whose tuples satisfy all the time-sensitive constraints. The procedure is based on the simplified topology generated by the pre-processing phase, and jointly considers routing and scheduling processes. The constraints and optimization objective of multicast time-sensitive networks are first discussed, and the RS procedure is then presented to synthesize a valid schedule.

**A. TIME-SENSITIVE CONSTRAINTS**

The time-sensitive constraints consist of the routing constraints, path-dependent constraints, contention-free constraints, resource constraints and latency constraints. The five constraints for flow $f_a$ are defined as follows:

1) **ROUTING CONSTRAINTS**

The routing constraints describe the transmission continuity for a flow over the network. Flow $f_a$ should be transmitted from the source $s_a$ to all the destinations $D_a$ through the relay nodes. The routing constraints are given as follows:

$$\forall (v_i, s_a) \in E : \sum_{(v_i, r_a) \in E} l_{(v_i, r_a), f_a} = 0, \sum_{(r_a, v_j) \in E} l_{(r_a, v_j), f_a} \geq 1 \quad (3)$$

Equation (3) describes that flow $f_a$ is generated from the source $s_a$. To ensure $s_a$ is the source node, there is at least one outgoing link of $s_a$ is occupied by flow $f_a$, and no incoming link is occupied.

$$\forall r_a \in V \setminus \{s_a\} \cup D_a : \sum_{(v_i, r_a) \in E} l_{(v_i, r_a), f_a} \leq 1,$$

$$\sum_{(v_i, r_a) \in E} l_{(v_i, r_a), f_a} \leq \sum_{(r_a, v_j) \in E} l_{(r_a, v_j), f_a},$$

$$\sum_{(r_a, v_j) \in E} l_{(r_a, v_j), f_a} \leq M \times \sum_{(v_i, r_a) \in E} l_{(v_i, r_a), f_a} \quad (4)$$

Equation (4) describes the forwarding behavior of all the possible relay nodes of flow $f_a$. As only relay nodes are involved in this equation, these constraints are not applied to the source and destinations of flow $f_a$. Since a multicast model is adopted, a relay node requires that if flow $f_a$ is transmitted through one incoming link of the node, there is at least one outgoing link that is occupied by the flow, and vice versa. It should be noted that no incoming and outgoing link mean that flow $f_a$ is not transmitted through the node.

As shown in Equation 4, the first inequation ensures that there is at most one incoming link of a relay node, maintaining a tree structure. The last two inequations formulate the contrapositive constraints of the forwarding behavior of a relay node, where no outgoing link is occupied means that flow $f_a$ is not transmitted through any of the incoming links of the node, and vice versa. As a relay node may have multiple outgoing links, $M$ is used to constrain the routing variables. $M$ is a sufficiently large integer, which is commonly used to formulate the conditional constraints in integer linear programming (ILP).

$$\forall d_a \in D_a : \sum_{(v_i, d_a) \in E} l_{(v_i, d_a), f_a} = 1, \sum_{(d_a, v_j) \in E} l_{(d_a, v_j), f_a} = 0 \quad (5)$$

Equation (5) describes that flow $f_a$ finally arrives at the Destinations $D_a$. To ensure $d_a \in D_a$ is a destination node,
there is exactly one incoming link of $d_a$ is occupied by flow $f_a$, and no outgoing link is occupied.

$$\forall [v_i, v_j] \in E : \quad \phi_{[v_i, v_j], f_a} \leq M \times \text{lo}_{[v_i, v_j], f_a} \quad (6)$$

Equation (6) describes the routing constraints for scheduling variables. As shown in this equation, the scheduling variable $\phi_{[v_i, v_j], f_a}$ is tied to zero if the link $[v_i, v_j]$ is not occupied by flow $f_a$.

$$\forall [s_a, v_i] \in E : \quad \phi_{[s_a, v_i], f_a} \geq \phi_{s_a} - M \times (1 - \text{lo}_{[s_a, v_i], f_a}),$$

$$\phi_{[s_a, v_i], f_a} \leq \phi_{s_a} + M \times (1 - \text{lo}_{[s_a, v_i], f_a}) \quad (7)$$

Equation (7) describes the timing constraints for the flow $f_a$. The path-dependent constraints describe the transmission time points of the source node. It is assumed that all the sending time points $\phi_{[s_a, v_i], f_a}$ of the source $s_a$ should be the same value, which is denoted by $\phi_{s_a} \in \mathbb{N}$. In addition, the introduction of $M$ ensures that the sending time point of $\phi_{[s_a, v_i], f_a}$ is equal to $\phi_{s_a}$ only when flow $f_a$ is transmitted via the link $[s_a, v_i]$.

2) PATH-DEPENDENT CONSTRAINTS

The path-dependent constraints describe the transmission timing constraints for a flow. As a flow can only be forwarded to the next switch after it has arrived at the current switch, the path-dependent constraints for the flow $f_a$ are given as follows:

$$\forall r_a \in V \setminus ([s_a] \cup D_a), \quad \forall v_j | [v_i, r_a] \in E, \forall v_j | [r_a, v_j] \in E :$$

$$\phi_{[r_a, v_j], f_a} = \phi_{[v_i, r_a], f_a} + \Delta t_a + \Delta t_{p_a} + M \times (\text{lo}_{[v_i, r_a], f_a} + \text{lo}_{[r_a, v_j], f_a} - 2) \quad (8)$$

where $[v_i, r_a]$, $[r_a, v_j]$ are two adjacent links in the network, $t_a$ is the transmission delay of flow $f_a$ over the link, and $\Delta t_{p_a}$ is the maximum process delay in each switch, which is closely related to hardware implementation. In addition, the third component of the right part of the inequality indicates that the constraints only work when flow $f_a$ is transmitted through the link $[v_i, r_a]$ and $[r_a, v_j]$.

3) CONTENTION-FREE CONSTRAINTS

The contention-free constraints describe the exclusive link constraints for all the links. As a link can only transmit one data flow at the same time, the resource constraints are given below:

$$\forall [v_i, v_j] \in E, \quad \forall f_a, f_b \in F, a \neq b, q \in \mathbb{Z}, r \in \mathbb{N} :$$

$$q \times \text{gcd}(p_a, p_b) + r = \phi_{[v_i, v_j], f_a} - \phi_{[v_i, v_j], f_b} \quad \wedge$$

$$t_b - M \times (2 - \text{lo}_{[v_i, v_j], f_a} - \text{lo}_{[v_i, v_j], f_b}) \leq r \leq$$

$$\text{gcd}(p_a, p_b) - t_a + M \times (2 - \text{lo}_{[v_i, v_j], f_a} - \text{lo}_{[v_i, v_j], f_b}) \quad (9)$$

where $\text{gcd}(p_a, p_b)$ function calculates the greatest common divisor (GCD) of flow period $p_a$ and $p_b$. The constraints for each flow pair imply that the transmission of the two flows does not conflict in the greatest common divisor of their periods. To simplify presentation and maintain constraints linear, two auxiliary variables $q$, $r$ are introduced, where $q$ and $r$ are the quotient and remainder of $(\phi_{[v_i, v_j], f_a} - \phi_{[v_i, v_j], f_b}) / \text{gcd}(p_a, p_b)$. It should be noted that $r$ is always a non-negative integer and $q$ may be negative if $\phi_{[v_i, v_j], f_a} \leq \phi_{[v_i, v_j], f_b}$. In the meantime, $r$ is also the minimal difference in the transmission time $\phi_{[v_i, v_j], f_a}$ and $\phi_{[v_i, v_j], f_b}$ of flow $f_a$ and $f_b$ over the link $[v_i, v_j]$. To ensure that the transmission of the two flows does not conflict, the minimal transmission difference $r$ must be larger than the transmission time of each flow. In addition, the introduction of $M$ ensures that the constraints only work when the link $[v_i, v_j]$ is occupied by both of the flows.

4) RESOURCE CONSTRAINTS

The resource constraints describe the storage usage constraints for switches. The number of available queues of each port for real-time flows in switch $v_i$ is denoted by $N_q(v_i)$, and the queue occupied by flow $f_a$ in switch $v_i$ is denoted by $Q_{[f_a, v_i]}$. As the number of queues is limited by $N_q(v_i)$, the first resource constraint of switch $v_m$ is defined as follows:

$$\forall v_i \in SW : 1 \leq Q_{[f_a, v_i]} \leq N_q(v_i) \quad (10)$$

Furthermore, in order to avoid the impact on delay and jitter caused by undetermined order of arriving flows in the queue, the flows are isolated in the queue. That is, the storage times of flows in the same queue do not overlap. To guarantee this feature, the second resource constraint of switch $v_m$ is defined as follows:

$$\forall f_a, f_b \in F, a \neq b :$$

$$Q_{[f_a, v_m]} = Q_{[f_b, v_m]} \Rightarrow$$

$$\phi_{f_a, [v_m, v_n]} = \phi_{f_b, [v_m, v_n]} \quad \text{mod} \text{gcd}(p_a, p_b) \quad (11)$$

where $f_a$ transmits from $[v_i, v_m]$ to $[v_m, v_n]$, and $f_b$ transmits from $[v_i, v_m]$ to $[v_m, v_n]$. The left side of the inequality is the minimal time that $f_b$ arrives before $f_a$ in the queue over the macro period, and the right side is the storage time of $f_b$ in the queue. If $f_a$ and $f_b$ are allocated to the same queue, $f_b$ is supposed to leave the queue before $f_a$ arrives.

5) LATENCY CONSTRAINTS

The latency constraints describe the application constraints that each flow should arrive at all its destinations within the deadline. The latency constraints are formalized as follows:

$$\forall d_a \in D_a : \left( \sum_{[v_i, d_a] \in E} \phi_{[v_i, d_a], f_a} \right) - \phi_{s_a} \leq d_{d_a} - t_a \quad (12)$$

where $d_{d_a}$ is the deadline, and $t_a$ is the transmission delay of flow $f_a$ over the link. As constrained by equation (6), there is only one nonzero value in $\sum_{[v_i, d_a] \in E} \phi_{[v_i, d_a], f_a}$, which is the actual sending time point of flow $f_a$ to the destination $d_a$. As defined in equation (7), $\phi_{s_a}$ is the sending time point of the source node. In order to provide latency guarantee, the difference in the sending time points of the source and destinations should be within the deadline.
B. OPTIMIZATION OBJECTIVE

Finally, the optimization objective of the RS procedure is defined as follows:

\[
    \min \left( B_{\text{max}} + \frac{1}{|E| |F|} \times \sum_{(i,j) \in E} \log_{i,j} f_a \right)
\]  

(13)

The primary objective is \( B_{\text{max}} \), which is the maximum bandwidth utilization over all the links after scheduling flow \( f_a \). This objective balances traffic load in the network, and thus increases the number of links that meet threshold \( B_T \) for link grouping. The secondary objective is the number of occupied links multiplied by a weighted factor, which normalizes the number and ensures the secondary objective is always less than 1. Since a multicast model is adopted, the routing constraints (3)(4)(5) only ensure that there are routes from the source to all the destinations without the consideration of loops. The secondary objective eliminates potential loops, ensuring that routes are a multicast tree structure. It should be noted that scheduling variables are not introduced in the optimization objective, as a real-time flow is only required to arrive at all its destinations within the deadline (not the sooner, the better). Moreover, the additional constraints of \( B_{\text{max}} \) are shown below:

\[
    \forall [i, j] \in E : B_{[i,j]} + B_{f_a} \times \log_{i,j} f_a \leq B_{\text{max}} 
\]

(14)

where \( B_{[i,j]} \) is the total bandwidth utilization of the link \([i, j]\) before scheduling the flow, and \( B_{f_a} \) is the bandwidth utilization of flow \( f_a \). These constraints ensure that \( B_{\text{max}} \) is the maximum bandwidth utilization over the links.

C. RS PROCEDURE

As the time-sensitive constraints and the optimization objective are described above, the RS procedure is devised to generate a valid schedule in this subsection. The RS procedure is based on the simplified topology generated by the pre-processing phase. For each link group or merged link, the RS procedure regards it as one single link. Specifically, given a grouped Network \( G' \), the existing flow set \( f_e \) and the new flow set \( f_n \) from dynamic applications, the RS procedure generates a valid schedule \( S_v \) for the new flow set \( f_n \) without disturbing the schedule of existing flow set \( f_e \). The pseudo-code of the procedure is shown in Procedure 2.

As the schedule synthesis problems are known to be NP-complete [12], the ILP solver is used in the RS procedure. The procedure iteratively selects a flow \( f_a \) from the new flow set \( f_n \) (line 2), and generates the constraint set \( C \) for this flow (line 3). Then, the ILP solver takes the constraint set \( C \) as the input, and tries to synthesize a schedule \( S_v \) for the flow \( f_a \) (line 4). If a valid schedule is synthesized, the schedule \( S_v \) and the flow \( f_a \) are added into the schedule \( S_v \) and the existing flow set \( f_e \) respectively (line 6). Otherwise, the procedure fails and returns an empty schedule (line 8). As the schedule is generated based on the simplified topology taking advantage of the link groups, the schedule synthesis procedure is sharply accelerated.

VII. POST-PROCESSING PHASE

In this section, the two steps, schedule mapping and splitting, in the post-processing phase are illustrated to handle the results of schedule synthesis.

A. SCHEDULE MAPPING

If a valid schedule \( S_v' \) is able to be synthesized for the grouped network \( G' \), \( S_v' \) needs to be transferred to a valid schedule \( S_v \) for the original network \( G \). To transfer from \( S_v' \) to \( S_v \), Lemma 1 is introduced below:

Lemma 1: If a schedule \( S_v' \) is a valid schedule for \( G' \), a valid schedule \( S_v \) can be constructed for \( G \).

Proof: Without loss of generality, a flow \( f_a \) is chosen in the schedule \( S_v' \). The routes of \( f_a \) in \( G \) from the source \( s_q \) are first constructed by the routes in schedule \( S_v' \). If a node \( v_i \) is connected to a link group \( r \) in \( S_v' \), the link \([v_i, v_j]\) in \( G \) that connects \( v_i \) and node \( v_j \) in \( r \) is added into the route. If a link group \( r_1 \) is connected to a link group \( r_2 \) in \( S_v' \), the link \([v_i, v_j]\) in \( G \) is added into the route where node \( v_i \) in \( r_1 \) is connected to node \( v_j \) in \( r_2 \). If a link group \( r \) is connected to \( v_j \) in \( S_v' \), the link \([v_i, v_j]\) in \( G \) that connects node \( v_i \) in \( r \) and \( v_j \) is added into the route. Then, the routes are complemented in the link groups to form a complete multicast tree. Finally, the scheduling results in \( S_v' \) are assigned to the corresponding links in \( S_v \). Since there is no transmission conflict in \( S_v' \) for \( G' \), there is no conflict in \( S_v \) for \( G \). Moreover, the latency constraints are satisfied for \( S_v \) according to \( S_v' \). Therefore, \( S_v \) is a valid schedule for \( G \). \( \Box \)

From the above, the schedule \( S_v' \) can be simply transferred to a valid schedule \( S_v \) for the original network \( G \) using the method in the proof of the Lemma 1.

B. SPLITTING

Although the grouped network \( G' \) accelerates the schedule synthesis procedure, it is an abstract of the original network \( G \). That is, there is not necessarily a valid schedule in the grouped network \( G' \), even though a valid schedule exists in the original \( G \). The reason is that the constraints are tightened where the scheduling variables are forced to not overlap in a link group or merged link. For example, if the total bandwidth utilization of a link group or merged link exceeds 1, the RS
The procedure is certain to fail to generate a valid schedule for the grouped network $G'$. When the RS procedure fails, the grouped network $G'$ needs to be refined to maintain schedulability. The splitting step analyzes the scheduling results, and locates the link group or merged link whose time-sensitive constraints are violated. Then, the link with the highest bandwidth utilization is split out of the link group or merged link, and is marked as not being considered for future link grouping or merging. Finally, the schedule synthesis step tries to generate a valid schedule based on the refined network $G''$.

VIII. EVALUATION

In this section, the performance of the AGRS approach is presented. The experiment setup is first discussed, and schedulability, bandwidth bottleneck, impact of $B_T$ and execution time are then evaluated. The experimental results are compared to the SMT method [10] with fixed routing (FR), which is generated by the traditional Steiner Tree [32]. Moreover, the results of the RS procedure without the pre-processing and post-processing phase are also compared.

A. EXPERIMENT SETUP

The experiments are performed on a hardware testbed as shown in Fig. 2. The testbed consists of TSN switches, end devices and a controller. A TSN switch is a fully featured 24-port real-time switch implemented by FPGA with a Xilinx Chip xc7vx690t of the Virtex-7 family, which supports on-the-fly reconfiguration of schedules by the controller through a southbound interface (SBI) similar to the OpenFlow protocol [33]. The testbed works in 100 Mbps.

In addition, a schedule synthesis tool is implemented for multicast time-sensitive networks, and the underlying ILP solver employed in this tool is IBM CPLEX (version 12.7.1.0). The controller is a Windows 10 (64 bit) desktop equipped with an Intel i7 4790K quad-core 4.00GHz and 32 GB memory. The scheduling approaches are evaluated via various network topologies generated by NetworkX, which is a popular Python package for the creation of complex networks. Four different network topologies are first created using the 5 model, each consisting of 24 switches, 128 core links and 72 end devices. Then, flows are randomly generated from 64 bytes to 1518 bytes with periods of 2, 4, 8, 16 and 32 ms and a deadline of 8 ms, which simulates the industrial control data. The source and destinations of each flow are randomly selected among the end devices. Finally, 1000 scenarios are generated by combining the four topologies and data transfer requirements for each case.

B. SCHEDULABILITY

To evaluate schedulability, flows are gradually added into networks until the scheduling procedure fails. For each number of flows, schedulability is calculated by dividing the number of schedulable scenarios from the total scenarios. As shown in Fig. 3, FR can schedule about 150 flows, while RS and AGRS can schedule about 250 flows. That is, schedulability is improved by about 68% through jointly considering routing and scheduling processes. The schedulability of FR drops dramatically after flow number exceeds 200, and FR can schedule no more than 323 flows. As a contrast, the schedulability of RS and AGRS is above 90%, which is still acceptable. Moreover, RS and AGRS can schedule at most about 450 flows, which is about 39% more than FR. It should be noted that there is a little difference of the schedulability between RS and AGRS. As the scheduling procedure schedules subsequent flows without disturbing previous flows, previous scheduling results may affect subsequent scheduling results. Since AGRS adopts a link grouping strategy, the assignment of scheduling variables over a specific link may be scattered to avoid conflict within link groups or merged links, while RS does not have this restriction. However, neither of the two strategies is significantly better than the other as shown in Fig. 3.

C. BANDWIDTH BOTTLENECK

It is observed that schedulability is closely related to the maximum bandwidth utilization over the links in the network. In addition, schedulability drops sharply after the maximum bandwidth utilization exceeds 60%. Therefore, the maximum bandwidth utilization is used to indicate bandwidth bottleneck. As shown in Fig. 4, FR has the highest maximum bandwidth utilization due to an adoption of fixed routes. Since RS includes load balancing in the optimization objective, the scheduling procedure can adaptively choose routes
according to the traffic load over the links. Therefore, bandwidth bottleneck is dramatically reduced compared with FR. Furthermore, although AGRS also uses a load balancing strategy, the traffic load of links within a link group or merged link is not balanced. Therefore, the bandwidth bottleneck of AGRS is about 30% higher than RS, which is the reason why the schedulability of AGRS drops before RS.

D. IMPACT OF $B_T$

As the link grouping step only merges complete subgraphs within the bandwidth threshold $B_T$, $B_T$ may have a large impact on the performance of the AGRS approach. In this subsection, the AGRS approach is evaluated under various values of $B_T$ as 0, 0.3, 0.6 and 1. If the value of $B_T$ is 0, grouping condition cannot be satisfied, and no complete subgraph is merged. Thus, the AGRS approach degrades to the RS approach. If the value of $B_T$ is 1, the complete subgraphs are always merged in the descending order of vertex number, and the adaptivity of the AGRS approach disappears.

Fig. 5 shows the impact of $B_T$ on the execution time of scheduling procedure. The horizontal axis presents the number of existing flows in the network, and the vertical axis presents the scheduling time per flow. Since the link grouping step merges complete subgraphs as much as possible when $B_T = 1$, the network topology is the most simplified among the four values. Therefore, the running time of $B_T = 1$ is the least when the number of flows is less than 50. As the number of flows in the network grows, the possibility of potential scheduling conflicts within the complete subgraphs increases dramatically. When scheduling conflicts occur, the schedule synthesis phase fails to generate a valid schedule based on the current simplified topology, and the splitting step is invoked to refine the topology. In this scenario, the schedule synthesis phase needs to be re-run based on the new topology, which consumes large execution time. As shown in Fig. 5, due to multiple refinement, the running time of $B_T = 1$ is even worse than the no-merging scenario ($B_T = 0$) when the number of flow exceeds 80. As the AGRS approach uses adaptive merging strategy, the running time performance of $B_T = 0.3$ and 0.6 is satisfying. Moreover, the performance of $B_T = 0.3$ is better than $B_T = 0.6$ when the number of flow exceeds 90.

E. EXECUTION TIME

Fig. 6 shows the execution time of scheduling procedure. The horizontal axis presents the number of existing flows in the network, and the vertical axis presents the scheduling time per flow. Since the link grouping step merges complete subgraphs as much as possible when $B_T = 1$, the network topology is the most simplified among the four values. Therefore, the running time of $B_T = 1$ is the least when the number of flows is less than 50. As the number of flows in the network grows, the possibility of potential scheduling conflicts within the complete subgraphs increases dramatically. When scheduling conflicts occur, the schedule synthesis phase fails to generate a valid schedule based on the current simplified topology, and the splitting step is invoked to refine the topology. In this scenario, the schedule synthesis phase needs to be re-run based on the new topology, which consumes large execution time. As shown in Fig. 5, due to multiple refinement, the running time of $B_T = 1$ is even worse than the no-merging scenario ($B_T = 0$) when the number of flow exceeds 80. As the AGRS approach uses adaptive merging strategy, the running time performance of $B_T = 0.3$ and 0.6 is satisfying. Moreover, the performance of $B_T = 0.3$ is better than $B_T = 0.6$ when the number of flow exceeds 90.
to have a similar running time performance compared to FR, and the execution time of AGRS is also in seconds when there are 100 flows in the network. Moreover, the execution time of AGRS is reduced by 74% though the link grouping strategy compared to RS as shown in Fig. 6. As industrial networks have rich connectivity due to redundancy, our link grouping strategy is general to accelerate the scheduling procedure.

IX. CONCLUSION
The time-sensitive networks are urgently desired in industrial automation to provide real-time control, and the time-triggered mechanism is a promising technology to deploy due to its determinism. However, time-triggered networks faces the challenges from dynamic applications in the trend of IIoT. The schedule synthesis approach should take fully advantage of the bandwidth without disturbing the existing schedule, and have low execution time to provide a fast response to the data transfer requirements. In this paper, a novel adaptive group routing and scheduling (AGRS) approach is proposed for multicast time-sensitive networks. In this approach, a pre-processing phase is first devised to accelerate the schedule synthesis procedure for dynamic applications, which prunes unnecessary links and adaptively groups qualified complete subgraphs of the topology. Then, a routing and scheduling (RS) procedure that jointly considers routing and scheduling processes is developed to improve schedulability. Experiments show that the AGRS approach can provide better schedulability and lower execution time compared to existing approaches.

REFERENCES
[1] M. Wollschlager, T. Sauter, and J. Jasperneite, “The future of industrial communication: Automation networks in the era of the Internet of Things and industry 4.0,” IEEE Ind. Electron. Mag., vol. 11, no. 1, pp. 17–27, Mar. 2017.
[2] R. Patzke, “Fieldbus basics,” Comput. Standards Inter., vol. 19, nos. 5–6, pp. 275–293, Oct. 1998.
[3] IEEE Standard for Ethernet, Standard IEEE 802.3-2015, 2016, pp. 1–4017.
[4] H. Kopetz, “The rationale for time-triggered Ethernet,” in Proc. Real-Time Syst. Symp., Barcelona, Spain, Nov. 2008, pp. 3–11.
[5] J. Jasperneite and J. Feld, “PROFINET: An integration platform for heterogeneous industrial communication systems,” in Proc. IEEE Conf. Emerg. Technol. Factory Autom., Catania, Italy, Apr. 2006, p. 8 and 822.
[6] S. G. Robertz, K. Nilsson, R. Henriksson, and A. Blomdell, “Industrial robot motion control with real-time Java and EtherCAT,” in Proc. IEEE Conf. Emerging Technol. Factory Autom. (EFTA 2007), Patras, Greece, Sep. 2007, pp. 1453–1456.
[7] P. Pedreiras, P. Gai, L. Almeida, and G. C. Bertazzo, “FTT-Ethernet: A flexible real-time communication protocol that supports dynamic QoS management on Ethernet-based systems,” IEEE Trans. Ind. Informat., vol. 1, no. 3, pp. 162–172, Aug. 2005.
[8] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer, “The time-triggered Ethernet (TTE) design,” in Proc. 5th IEEE Int. Symp. Object-Oriented Real-Time Distrib. Comput. (ISORC), Seattle, WA, USA, Jun. 2005, pp. 22–33.
[9] IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 25: Enhancements for Scheduled Traffic, Standard IEEE 802.1 Qbv, 2016, pp. 1–57.
[10] W. Steiner, “An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks,” in Proc. 31st IEEE Real-Time Syst. Symp., San Diego, CA, USA, Nov. 2010, pp. 375–384.
[11] S. S. Craciunas and R. S. Oliver, “Combined task- and network-level scheduling for distributed time-triggered systems,” Real-Time Syst., vol. 52, no. 2, pp. 161–200, Mar. 2016, doi: 10.1007/s11241-015-9244-x.
[12] Z. Hanzalík, P. Burget, and P. Sucha, “Proﬁnet IO IRT message scheduling with temporal constraints,” IEEE Trans. Ind. Informat., vol. 6, no. 3, pp. 369–380, Aug. 2010, doi: 10.1109/TII.2010.2052819.
[13] S. Jeschke, C. Brecher, T. Meisen, D. Ozdemir, and T. Eschert, “Industrial Internet of Things and cyber manufacturing systems,” in Industrial Internet of Things. Cham, Switzerland: Springer, 2017, pp. 3–19.
[14] E. Frontoni, J. Loncarsi, R. Pierdicca, M. Bernardini, and M. Sasso, “QoS-aware physical infrastructure for industry 4.0: Towards real-time virtual reality in smart manufacturing,” in Proc. Augmented Reality, Virtual Reality, Comput. Graph.-5th Int. Conf. (AVR), Otranto, Italy, Jun. 2018, pp. 422–434, doi: 10.1007/978-3-319-95282-6_31.
[15] R. Serna Oliver, S. S. Craciunas, and G. Stoger, “Analysis of deterministic Ethernet scheduling for the industrial Internet of Things,” in Proc. IEEE 19th Int. Workshop Comput. Aided Modeling Design Commun. Link Netw. (CAMAD), Greece, Dec. 2014, pp. 320–324, doi: 10.1109/CAMAD.2014.7033258.
[16] R. Serna 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. (RTAS), Porto, Portugal, Apr. 2018, pp. 13–24, doi: 10.1109/RTAS.2018.00008.
[17] R. Queck, “Analysis of Ethernet AVB for automotive networks using network calculus,” in Proc. IEEE Int. Conf. Veh. Electron. Softw. (ICVES), Jul. 2012, pp. 61–67, doi: 10.1109/ICVES.2012.6294261.
[18] L. Zhao, P. Pop, Q. Li, J. Chen, and H. Xiong, “Timing analysis of rate-constrained traffic in TTEthernet using network calculus,” Real-Time Syst., vol. 53, no. 2, pp. 254–287, Mar. 2017, doi: 10.1007/s11241-016-9265-0.
[19] H. Kopetz and G. Bauer, “The time-triggered architecture,” Proc. IEEE, vol. 91, no. 1, pp. 112–126, Jan. 2003.
[20] H. Kopetz, C. El Salloum, B. Huber, R. Obermaisser, and C. Paukovits, “Towards an integrated and distributed system for Ethernet-based RT systems,” in Proc. IEEE Int. SOC Conf., Newport Beach, CA, USA, Sep. 2008, pp. 87–90, doi: 10.1109/SOCF.2008.4641485.
[21] J. Ferreira, L. Almeida, J. A. Fonseca, P. Pedreiras, E. Martins, G. Rodriguez-Navas, J. Rigo, and J. Proenza, “Combining operational flexibility and determinability in FTTCAN,” IEEE Trans. Ind. Informat., vol. 2, no. 2, pp. 95–102, May 2006, doi: 10.1109/TII.2005.875508.
[22] A. Pop, P. Pop, P. Eles, Z. Peng, and A. Andrei, “Timing analysis of the FlexRay communication protocol,” Real-Time Syst., vol. 39, nos. 1–3, pp. 205–235, Aug. 2008, doi: 10.1007/s11241-007-9040-3.
[23] F. Pozo, W. Steiner, G. Rodriguez-Navas, and H. Hansson, “A decomposi- tion approach for SMT-based schedule synthesis for time-triggered net- works,” in Proc. IEEE 20th Conf. Emerg. Technol. Factory Autom. (ETFA), Luxemburg, U.K., Sep. 2015, pp. 1–8.
[24] F. Pozo, G. Rodriguez-Navas, W. Steiner, and H. Hansson, “Period-aware segmented synthesis of schedules for multi-hop time-triggered net- works,” in Proc. IEEE 22nd Int. Conf. Embedded Real-Time Comput. Symp. Appl. (RTCSA), Daegu, South Korea, Aug. 2016, pp. 170–175.
[25] S. S. Craciunas, R. S. Oliver, M. Chmelik, and W. Steiner, “Scheduling real-time communication in IEEE 802.1Qbv time sensitive networks,” in Proc. 24th Conf. Int. Conf. Real-Time Netw. Syst. (RTNS), Brest, France, 2016, pp. 183–192, doi: 10.1109/2997465.2997470.
[26] 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.
[27] F. Dür and N. G. Nayak, “No-wait packet scheduling for IEEE time-sensitive networks (TSN),” in Proc. 24th Int. Conf. Real-Time Netw. Syst. (RTNS), Brest, France, 2016, pp. 203–212, doi: 10.1109/2997465.2997494.
[28] E. Scheweigeth, P. Daniell, T. Timmermann, H. Parzyjega, and G. Mühl, “ILP-based joint routing and scheduling for time-triggered networks,” in Proc. 25th Int. Conf. Real-Time Netw. Syst. (RTNS), Grenoble, France, 2017, pp. 8–17, doi: 10.1109/3139258.3139289.
[29] F. Smirnov, M. Glaß, F. Reimann, and J. Teich, “Optimizing message routing and scheduling in automotive mixed-criticality time-triggered net- works,” in Proc. 54th Annu. Design Autom. Conf. (DAC), Austin, TX, USA, 2017, pp. 48:1–48:6, doi: 10.1145/3061639.3062298.
[30] N. G. Nayak, F. Durr, and K. Rothermel, “Incremental flow scheduling and routing in time-sensitive software-defined networks,” IEEE Trans. Ind. Inform., vol. 14, no. 5, pp. 2066–2075, May 2018, doi: 10.1109/TII.2017.2782235.

[31] E. A. Akkoyunlu, “The enumeration of maximal cliques of large graphs,” SIAM J. Comput., vol. 2, no. 1, pp. 1–6, Mar. 1973, doi: 10.1137/0202001.

[32] L. Kou, G. Markowsky, and L. Berman, “A fast algorithm for Steiner trees,” Acta Inf., vol. 15, no. 2, pp. 141–145, 1981, doi: 10.1007/BF00288961.

[33] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “OpenFlow: Enabling innovation in campus networks,” SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, p. 69, Mar. 2008, doi: 10.1145/1355734.1355746.

QINGHAN YU (Student Member, IEEE) received the B.S. degree in computer software from Tsinghua University, Beijing, China, in 2014, where he is currently pursuing the Ph.D. degree in software engineering with the School of Software.

His current research interests include real-time systems, network scheduling, and embedded systems.

MING GU received the B.S. degree in computer science from the National University of Defense Technology, Changsha, China, in 1984, and the M.S. degree in computer science from the Chinese Academy of Sciences, Beijing, China, in 1986.

She is currently a Professor with the School of Software, Tsinghua University, Beijing. Her research interests include software formal methods, software trustworthiness, and middleware technology.

* * *