IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 25, 2019, accepted January 18, 2020, date of publication January 29, 2020, date of current version February 6, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2970266

Enhancing Real-Time and Determinacy for Network-Level Schedule in Distributed Mixed-Critical System

JUN LU, HUAGANG XIONG, FENG HE, AND RONGWEI WANG
School of Electronic and Information Engineering, Beihang University, Beijing 100191, China
Corresponding author: Feng He (robinleo@buaa.edu.cn)

ABSTRACT Time-triggered Ethernet (TTE) is designed as a deterministic real-time network for mixed-critical real-time systems, such as industrial automation, aerospace, and aviation. Due to asynchrony between tasks and the network in end systems (ESes), time-triggered (TT) messages have to spend uncertain time in waiting for being scheduled after they are generated. The waiting time uncertainly increases end to end delays for TT messages and will degrade the real-time and determinacy of TT messages sequentially. The contribution of this paper is to present a new architecture SDTTE to minimize the end to end delays, so as to enhance real-time and determinacy of TT messages. More specifically, a frame-based network model is built to describe distributed network resources. Under the network model, the end to end delay model with asynchrony between tasks and the network is analyzed. To match the generated time with the triggered time for TT messages, the triggered mechanism of the TT traffic in ESes is regarded as event strategy to detect the generated time of TT messages automatically. Based on this, the software defined TTE (SDTTE) is presented to optimize TT schedule online in switches. Furthermore, a simplified algorithm based on Satisfiability Modulo Theories (SMT) is proposed to satisfy real-time computing requirements. Finally, experiments with three network sizes verify the availability of SDTTE and analyze its performance. The results show that end to end delays for TT messages in SDTTE are decreased by about 95% compared with those in TTE. And the delays for rate-constrained and best-effort messages in SDTTE are almost as well as those in TTE. The processing time is less than 10s. In general, SDTTE provides a method to optimize end to end delays for TT messages in TTE, thus SDTTE has more determinacy and real-time than TTE. Meanwhile, SDTTE makes dynamic reconfiguration possible in practice.

INDEX TERMS End to end delay, determinacy, real-time, software defined network, time-triggered ethernet.

I. INTRODUCTION

A. TIME-TRIGGERED SCHEDULE TABLE

Network is a core technology of architectural concepts tailored for deterministic operation, clean interfacing and predictable resource sharing in integrated fault-tolerant systems with time-, mission- and safety-critical functions. Time-triggered ethernet (TTE), as a deterministic networking solution based on 802.3 Ethernet, adds time-triggered (TT) mechanism to eliminate the transmission indeterminacy of the TT traffic [1]–[5]. As prerequisite, a global TT schedule table is required to avoid collision among different time-triggered windows [6].

The associate editor coordinating the review of this manuscript and approving it for publication was Giovanni Merlino.

When an avionics system maintains its mission pattern, the coresponding end systems (ESes) will also hold their task states, the generated time of a TT message in ES is usually a fixed value. A static global TT schedule table can meet time-triggered communication requirements. However, when an end system changes its task states or reconfigures, the generated time of a TT message may be changed as task is executed in different mission model. It needs some static global TT schedule tables that correspond to different missions. Whatever the state of the avionics system is, global TT schedule tables are static. Focus on the static schedule table design, various heuristic TT scheduling algorithms have been researched widely to obtain a feasible TT schedule table with different calculation complexity [7]–[9].

As the research further develops, Satisfiability Modulo Theories (SMT) is presented to solve the satisfiability of logic formulas [10], [11]. Naturally, TT schedule suits the SMT problem well since periodic frames can be represented by finite sets which repeat infinitely with a given period [12]. SMT was introduced into planning TT schedule to obtain a feasible TT schedule table easily [13]–[15].

However, the queuing delay for a TT message in an end system (ES) has not been considered into the conventional end to end delay [16]–[18]. Indeed, a TT frame might have to wait a full transmission period in the ES before being scheduled out [19]. Namely, the TT traffic introduces another delay from TT schedule due to inconsistency between its generated time and its triggered time. The range of the delay is from zero to a full transmission period. It might lead that the rate-constrained (RC) traffic has better real-time and determinacy than the TT traffic in some cases of low bandwidth utilization.

B. RELATED WORK

Hard real-time systems add synchronous model of computation and communication (MOCC) support [20]. Namely, a hard real-time system is a time-triggered architecture (TTA), where tasks, partitions, and network are synchronous. So the queuing delays in end systems usually are a constant value (e.g. millisecond-level), thus the real-time of the TT traffic has a dramatic decline. Meanwhile, in the case of reconfiguration, TT messages have to wait different queuing delays. It may lead a dramatic scheduling jitter for the TT traffic. Many researchers tried to synthesize tasks and the network to obtain an integrated TT schedule table [21]–[23]. The generated time of TT messages is known in hard real-time systems, the gap between the generated time and the triggered time can be used as a constraint to optimize TT schedule table.

In hard real-time systems, all tasks must schedule during the special time. Fraboul's team thinks that the loosely time-triggered architecture (L-TTA) is better in practice [20]. In addition to time-triggered, L-TTA can also realize some uncertain scheduling mechanisms, such as multi-priority in RC traffic. In L-TTA, TT-tasks continue being performed by static task schedule to guarantee the periodicity of TT-tasks. But system and network have their own synchronization mechanism, respectively. So task-level time and network-level time are inconsistent. It becomes difficult to predict the accurate generated time based on network-level time. It varies with task execution and can be different in each startup of an ES. How to obtain the accurate generated time of TT frames is the key problem on TT schedule table design.

Meanwhile, task-level time and network-level time are dynamically changing. So time drift between the generated time and the triggered time is also dynamical. It leads a jitter for the TT traffic. There must be an online detection mechanism to keep the TT traffic from a dramatic scheduling jitter. To solve this problem, it needs a centralized controller, mixed-critical network to support TT schedule online. McKeown presented software defined network (SDN) to dynamically adjust network configuration on campus network [24]. It makes centralized control possible. Thereafter, SDN has a considerable development in the mixed-critical network. Sampigethaya analyzed opportunities and challenges of SDN in aviation qualitatively [25]. SDN has huge advantages in the real-time network. Based on this, Heise analyzed performance evaluation of SDN hardware for avionic networks with avionics full duplex switched ethernet (AFDX) through the experiment [26]. However, the flow tables in switches are configured statically during the initial only, the advantage of dynamic adjustment in SDN is not taken. Further, Guck established network models in SDN-based real-time network [27]. The combination of a routing procedure and a network model needs to be further analyzed. Recently, Kumar introduced SDN to guarantee end-to-end network delay, but they only consider the optimization problem for RC traffic [28]. How to realize dynamic TT schedule by SDN is another critical problem to guarantee determinacy online.

Scheduling algorithm determines the matching accuracy between the tasks schedule and the network schedule. The above SMT-based algorithms can obtain a feasible solution, but they can not solve optimal problem. Craciunas presented an optimized SMT-based methods to suit online scheduling scenarios [29]. Thereafter, Craciunas transformed the task- and network-level schedule co-synthesis into a Mixed Integer Programming (MIP) problem with different objectives [12]. But he found that the MIP spends too much time to perform calculating online, and the processing time of the optimized SMT-based method is still too long. According to the processing time and matching accuracy of the tasks schedule and the network schedule, how to optimize the SMT-based algorithm is the third problem to guarantee task-level real-time and determinacy requirement.

C. CONTRIBUTIONS

To solve the above three problems, we propose the software defined time-triggered ethernet (SDTTE) framework which decreases the end to end delay and jitter for TT messages caused by asynchrony between tasks and the network and can increase real-time and determinacy of the TT traffic. The primary contributions of this paper are summarized as follows.

1) A triggered mechanism of the TT traffic, which makes the generated time of each TT message indirectly detectable, is presented to ease asynchrony between tasks and the network in end systems (ESes). The triggered mechanism of the TT traffic is regarded as event-triggered type in the source ESes. Meanwhile, the TT traffic keeps time-triggered mechanism among switches.
2) A SDN-based TTE network architecture SDTTE is proposed to guarantee online synchronization between tasks and the network, when variable mission and reconfiguration occur. The SDTTE can detect the generated time of each TT frame and plan TT schedule table for all switches online.

3) A simplified SMT-based algorithm is presented to reduce the processing time of TT schedule with synchronization between tasks and the network.

D. PAPER ORGANIZATION

The remainder of this paper is organized as follows. Section 2 builds network model and the end to end delay model. Subsequently, section 3 presents a new message model by changing the triggered mechanism of the TT traffic to make the generated time of TT frames detectable. And then SDTTE is proposed to synchronize tasks with the network online. Meanwhile, to reduce the processing time of TT schedule, a simplified algorithm based on SMT is presented. Section 4 verifies the availability of SDTTE by using a simulation experiment based on OMNeT++. Subsequently, the performance of messages in SDTTE is compared with those in TTE by statistical experiments in middle and large mesh network. Meanwhile, some phenomena in the experiments are described and illustrated. Finally, some conclusions are given in section 5.

II. SYSTEM MODEL

A. NETWORK MODEL

A TTEthernet network is a full duplex switched network with time-triggered and event-triggered mechanism. A TTE network can be formally modeled as a directed graph $G(V, E)$, where the set of vertices V comprises the communication nodes (switches and end-systems) and the edges E represent the directional communication links between nodes, similar to Zhao *et al.* [18]. For an edge $e_{i,j}$ with the bandwidth $C_{e_{i,j}}$ (e.g. 100 Mbit/s, 1 Gbit/s, etc.), it can be given by

$$e_{i,j} = [v_i, v_j] \in E, \quad v_i, v_j \in V \tag{1}$$

We model time-triggered communication via the concept of virtual link (VL), where a virtual link is a logical data-flow path in the network from a sending node to one or several receiving nodes. A typical virtual link $vl_p \in VL$ from a producer task running on ES v_{e1} to a consumer task running on ES v_{e2}, routed through the switches $v_1, \ldots, v_n$ can be described as follows, similar to Steiner [14].

$$\begin{aligned} vl_p &= [v_{e1}, v_1][v_1, v_1][v_1, v_2] \cdots [v_n, v_n][v_n, v_{e2}] \\ &= e_{e1,1}e_{1,1}e_{1,2} \cdots e_{n,n}e_{n,e2} \end{aligned} \tag{2}$$

where $e_{n,n}$ denotes a frame delivering in a switch.

Let M denote the set of all messages in the system. We model a message $m_p \in M$ associated with the virtual link vl_p by the tuple.

$$m_p =< P_p, L_p, vl_p > \tag{3}$$

where P_p is the period and L_p is the size in bytes. $m_{p,q}$ can represent the q^{th} instance of the message m_p.

$$m_{p,q} =< T_{p,q}, P_p, L_p, vl_p > \tag{4}$$

where $T_{p,q}$ is the generated time of the message instance $m_{p,q}$ in an ES.

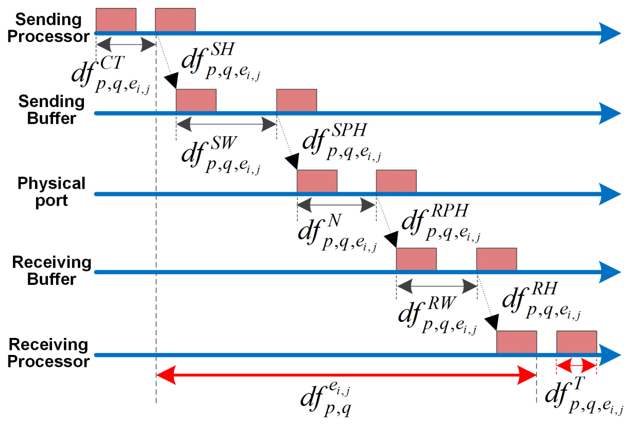


FIGURE 1. All possible delays of a frame between two nodes.

A frame $f^{e_{i,j}}_{p,q}$ uniquely corresponds to a message instance $m_{p,q}$ scheduled on a particular link $e_{i,j}$. Let F be the set of all frames $f^{e_{i,j}}_{p,q}$ in the system. $F^{e_{i,j}}_p \in F$ denotes the ordered set of all frames $f^{e_{i,j}}_{p,q}$ carried by virtual link vl_p scheduled on physical link $e_{i,j}$ and the sequence of all frames is decided by frame offsets. Furthermore, we denote the first and last frame in the set $F^{e_{i,j}}_p$ with $f^{e_{i,j}}_{p,1}$ and $f^{e_{i,j}}_{p,end}$, respectively.

A frame $f^{e_{i,j}}_{p,q} \in F^{e_{i,j}}_p$ is defined by the following tuple, similar to Steiner [14].

$$f^{e_{i,j}}_{p,q} =< of^{e_{i,j}}_{p,q}, pf^{e_{i,j}}_{p,q}, lf^{e_{i,j}}_{p,q}, cf^{e_{i,j}}_{p,q} > \tag{5}$$

where $of^{e_{i,j}}_{p,q}$ is the offset of the frame $f^{e_{i,j}}_{p,q}$, $pf^{e_{i,j}}_{p,q}$ is its period, $lf^{e_{i,j}}_{p,q}$ is the transmission time, and $cf^{e_{i,j}}_{p,q}$ is the bandwidth of physical links $e_{i,j}$. For a frame $f^{e_{i,j}}_{p,q}$, it typically has

$$pf^{e_{i,j}}_{p,q} = P_p \tag{6}$$

$$lf^{e_{i,j}}_{p,q} = L_p / cf^{e_{i,j}}_{p,q} \tag{7}$$

B. END TO END DELAY MODEL

A frame can be regard as the basic unit of an instance of a message on a particular link. For a frame $f^{e_{i,j}}_{p,q}$, all possible delays can be illustrated in Fig. 1, where $df^{CT}_{p,q,e_{i,j}}$ denotes computational delay in a ES processor or technical delay in a switch processor, $df^{SH}_{p,q,e_{i,j}}$ denotes the transmission delay between the processor and the sending buffer, $df^{SW}_{p,q,e_{i,j}}$ is the queuing delay of the message waiting to be sent in the sending buffer within a node, $df^{SPH}_{p,q,e_{i,j}}$ is the communication delay between the sending buffer and the physical port, $df^{N}_{p,q,e_{i,j}}$ denotes the communication delay on physical link $e_{i,j}$, $df^{RPH}_{p,q,e_{i,j}}$ is the communication delay between the physical port and the receiving buffer, $df^{RW}_{p,q,e_{i,j}}$ is the queuing delay of the message waiting to be received in the receiving buffer within a node, $df^{RH}_{p,q,e_{i,j}}$ denotes the communication delay between the receiving buffer and the receiving processor, $df^{T}_{p,q,e_{i,j}}$ denotes transmitting delay. Usually, it can be given by

$$df^{T}_{p,q,e_{i,j}} = lf^{e_{i,j}}_{p,q} \tag{8}$$

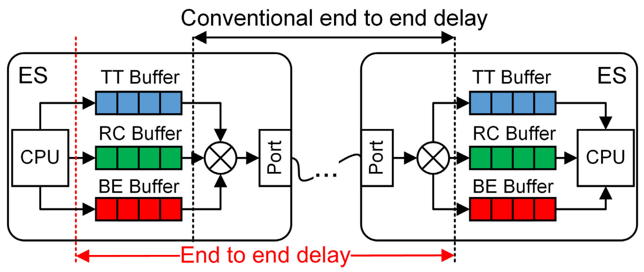


FIGURE 2. **End to end delay description.**

As the example in Fig. 1 shows, a frame delay is defined as $df_{p,q}^{e_{i,j}}$.

$$df_{p,q}^{e_{i,j}} = \begin{cases} df_{p,q,e_{i,j}}^{CT}, & i = j \\ \sum\limits_{z \in Z_a} z, & i \neq j \end{cases} \tag{9}$$

where Z_a is the set of all the possible transmission delays in Fig. 1.

A message instance is the set of frames, its delay can be derived by delays of frames. Conventional end to end delay, namely the network-level transmission delay, begins from the time that a message is sent out from a sending buffer in the corresponding ES and ends the time that the last bit is received from the physical link by the destination node. Thus, the conventional end to end delay for a message instance can be given by

$$D_{m_{p,q}}^{c} = \sum_{e_{i,j} \in vl_p} (df_{p,q}^{e_{i,j}} + df_{p,q,e_{i,j}}^{CT}) - \sum_{z \in Z_c} z \tag{10}$$

where $Z_c = \{df_{p,q,e_{e1,1}}^{CT}, df_{p,q,e_{e1,1}}^{SH}, df_{p,q,e_{e1,1}}^{SW}, df_{p,q,e_{e1,1}}^{RH}, df_{p,q,e_{e1,1}}^{RW}\}$ is the set of some transmission delays in Fig. 1.

The conventional end to end delay and the end to end delay are shown in Fig. 2. For distributed mixed-critical system, the task and network are asynchronous. The network schedule table based on conventional end to end delay might keep a time-triggered frame waiting for a full transmission period in the end system. The end to end delay must contain the source queuing delay. Meanwhile, the network schedule table is not related to the destination queuing delay. So the end to end delay should not include the destination queuing delays for optimizing network schedule table.

In order to describe the entire delay for TT message in the network scheduling, we define the end to end delay beginning from the generated time point, which is simialr as Craciunas and Oliver [12]. Thus the end to end delay for a message instance is given by

$$D_{m_{p,q}}^{c} = \sum_{e_{i,j} \in vl_p} (df_{p,q}^{e_{i,j}} + df_{p,q,e_{i,j}}^{CT}) - \sum_{z \in Z} z \tag{11}$$

where $Z = \{df_{p,q,e_{e1,1}}^{CT}, df_{p,q,e_{e1,1}}^{RH}, df_{p,q,e_{e1,1}}^{RW}\}$ is a set of some transmission delays in Fig. 1.

The end to end delay for a message m_p can be derived by the max end to end delay for message instances $m_{p,q}$.

$$D_{m_p} = \max_{q=1,\ldots,n} D_{m_{p,q}} \tag{12}$$

III. SDTTE

A. PRELIMINARY BACKGROUND

For each message in the TTE, there are three traffic classes, TT traffic, RC traffic and best effort (BE) traffic.

1) A time-triggered (TT) traffic needs a global schedule table which defines the triggered time when frames should be sent. A global synchronized clock is required to support the global schedule table.
2) A rate-constrained (RC) traffic has two key parameters: a Bandwidth Allocation Gap (BAG) which is the minimum time interval between two consecutive frames of a RC traffic in the source ES and a maximal frame size which limits the frame length. The constraints are the same as AFDX.
3) A best-effort (BE) traffic is simply a class for low-priority Ethernet traffic without time-triggered and BAG guarantees, but the timing behavior should pay respect to memory and frame delay constraints [30]–[32].

Though the three type traffics have different schedule strategy, the most delays shown in Fig. 1 are the same or similar, except the queuing delay caused by triggered mechanism. As RC traffic and BE traffic are both event-triggered, there is no fixed triggered time for their message transmission so that the queuing delay of the sending buffer in the source ESes is usually quite short when considering messages in the corresponding ES could not be generated at the the same time. But they are probably blocked by other messages when they are forwarded across every switch along their path due to asynchronous message arrival from different physical links or network segments. The TT traffic is time-triggered, so there is no blocked delay when it is forwarded across every switch according to predesigned schedule table. But a TT message has queuing delay in the sending buffer which is related to the generated time and the triggered time.

B. MESSAGE MODEL

Due to asynchrony between tasks and the network, usually the triggered time of TT messages does not match with the generated time so that the queuing delays of the sending buffer is uncertain. If the generated time of TT messages is known, the triggered time in TT schedule table can be designed to match with the generated time. It leads to sharply reduce the queuing delays of the sending buffer for TT messages so that the real-time and determinacy are improved. In practice, it is impossible to measure the generated time of TT messages as it varies with time drift between system time and network time in each startup or reconfiguration.

In fact, the dynamic generated time of TT messages is so difficult to measure that we have to take a roundabout

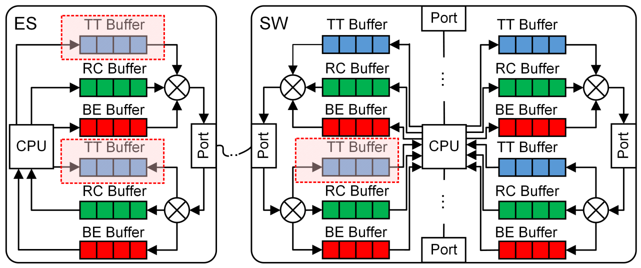


FIGURE 3. New architecture of an ES without sending buffer and receiving buffer for TT.

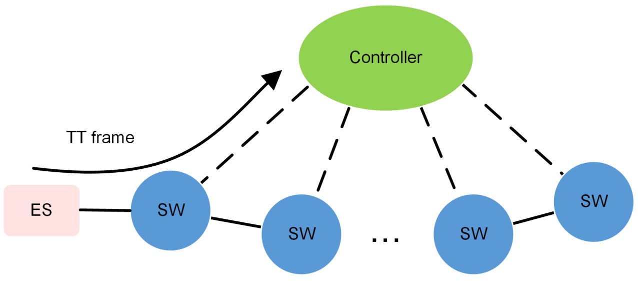


FIGURE 4. The network architecture of Software Defined TTE.

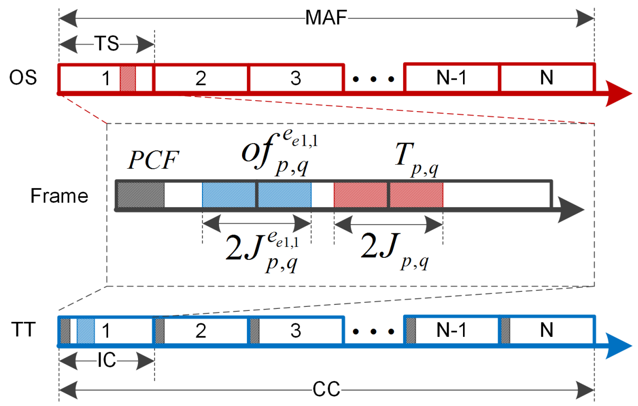


FIGURE 5. The asynchrony between tasks and the network. (Cluster cycle (CC) is the lowest common multiple (LCM) of all message periods and integration cycle (IC) is the greatest common divisor (GCD) of all message periods. PCF is the protocol control frame for clock synchronization in TTE, and the major time frame (MAF) is defined as the least common multiple of all partition periods in the corresponding ES. In each major time frame, there are several identical partition scheduling windows. TS denotes time slicing.)

method. Our method does not change the frame format of the TT messages. Only the TT buffers in end systems and ingresses of edge switches, which are the red shaded components as shown in Fig. 3, are disabled. So TT traffic is delivered by using event-triggered mechanism in end systems and ingresses of edge switches. As the highest priority traffic they are, TT traffic can be regarded as RC traffic with the highest priority in end systems and ingresses of edge switches. TT traffic still keeps time-triggered mechanism in other switches.

TT frames in ESes do not have to be sent/received during the predesigned sending/receiving time window. Instead, TT frames can be sent from the source ESes as soon as they are generated, and they can be received by the destination ESes as soon as they arrive at the receiving ports. In the same way, TT frames from the source ESes can be also received by switches as soon as they arrive at the receiving ports. The receiving time of TT messages from ESes to switches can indirectly reflect the generated time of TT messages. So the problem about detecting the generated time of TT messages can be solved in the first connected switch from the source ES for the TT traffic.

C. NETWORK ARCHITECTURE

To measure the dynamic received time of TT messages from ESes to switches, software defined network is introduced into TTE as shown in Fig. 4. TTE switches must have the ability of software defined path. Namely, the flow table in TTE switches can be configured by a central controller.

A TT frame is sent to the connected switch as soon as the corresponding TT message is generated. Although the TT traffic is regarded as event-triggered type in all ESes, it is time-triggered in all switches. TT schedule table is still required for switches.

When the system need be reconfigured, there is no flow table or a huge difference between the generated time and the triggered time, the switch would report the TT message to a centralized controller so that the centralized controller can obtain the received time of the TT message from the source ES to the first connected switch. And then the generated time of the TT message can be calculated indirectly by the centralized controller. By using this method, we can get the practical generated time of TT messages, it also provides a basis to further optimize the schedule table.

When the system need not be reconfigured, we defined a threshold and time drift which are relative to the first generated time for a time-triggered message after last update. If the time drift exceeds the threshold, the network schedule table will be reconfigured online. Calculating a time-triggered schedule table, which is a NP-complete problem, needs a long time consumption [12]. So the threshold avoids big jitters for time-triggered messages during updating the online time-triggered schedule table. The reason for selecting the threshold is that time-triggered messages may update their schedule table simultaneously so that resource consumption of scheduling solution is reduced, when system and network have their own synchronization mechanisms.

D. TT SCHEDULING CONSTRAINTS

To get the optimal TT schedule table, the constraints must be suitable. Since TT messages are not triggered by time in ESes, there is no ES constraint for TT schedule table, not like Craciunas and Oliver [12].

For $\forall v_i \neq v_j \neq v_k \in V, m_p \neq m_{p1} \neq m_{p2} \in M, f_{p,q} \neq f_{p,q1} \neq f_{p,q2} \in F$, scheduling constraints of the end to end delay are illustrated as follows.

1) BOUND CONSTRAINT

To plan scheduling table, we need only program offset time of all time-triggered frames during a cluster cycle t_c as shown in Fig. 5. These offsets $of_{p,q}^{e_{i,j}}$ should be within the scope of cluster cycle t_c along virtual links. Namely, we need only limit that the difference between the maximum offset and minimum offset in an edge $e_{i,j}$ is not greater than the cluster cycle t_c as follows. Because these offsets are relative value based on a discretionary reference point.

$$max(of_{p,q}^{e_{i,j}}) - min(of_{p,q}^{e_{i,j}}) < t_c \tag{13}$$

2) COLLISION-FREE CONSTRAINT

Any frames (e.g. $of_{p1,q1}$, $of_{p2,q2}$) can not collide with each other on the link $e_{i,j}$ at the same time. They need to satisfy with inequality (14) or inequality (15) which just separates the related frames according to different offsets.

$$of_{p1,q1}^{e_{i,j}} + lf_{p1,q1}^{e_{i,j}} < of_{p2,q2}^{e_{i,j}} \tag{14}$$

$$of_{p2,q2}^{e_{i,j}} + lf_{p2,q2}^{e_{i,j}} < of_{p1,q1}^{e_{i,j}} \tag{15}$$

3) TIME SERIES CONSTRAINT

The receiving time for a message instance $m_{p,q}$ on the physical link $e_{i,j}$ is less than the sending time on the next physical link $e_{j,k}$ according to the required transmission time.

$$of_{p,q}^{e_{i,j}} + lf_{p,q}^{e_{i,j}} + d_{Innerhop} \le of_{p,q}^{e_{j,k}} \tag{16}$$

where $d_{Innerhop}$ is the technical delay in a switch.

4) PERIODIC MESSAGE CONSTRAINT

For any frames belonging to the same message m_p on physical link $e_{j,k}$, their offsets should have fixed interval due to message periodicity. For example, the interval between the $q2^{th}$ frame and the $q1^{th}$ frame should be $q2 - q1$ times period.

$$of_{p,q1}^{e_{i,j}} = of_{p,q2}^{e_{i,j}} + (q2 - q1)pf_{p,q2}^{e_{i,j}} \tag{17}$$

5) MAX DELAY CONSTRAINT

The end to end delay should not exceed the message deadline. In a general way, the message deadline is the message period $pf_{p,q}^{e_{e1,j}}$. Namely, the end to end delay should not exceed the message period $pf_{p,q}^{e_{e1,j}}$.

$$of_{p,q}^{e_{j,e2}} + lf_{p,q}^{e_{e1,j}} - of_{p,q}^{e_{e1,j}} < pf_{p,q}^{e_{e1,j}} \tag{18}$$

6) OFFSET CONSTRAINT

The offset in the first connected switch must be larger than the generation time to ensure the arrival TT frame can find an appropriate triggered window to be scheduled out from queuing buffer. Also the offset should not be too large to avoid a long time waiting. So it is the key parameter which should be calculated accurately as shown in Fig. 5.

Considering the synchronization precisions of the tasks schedule and the network schedule (e.g. $J_{p,q}$ and $J_{p,q}^{e_{e1,1}}$), the offset constraint can be described as follows.

$$of_{p,q}^{e_{e1,j}} > T_{p,q} + J_{p,q}^{e_{e1,1}} + J_{p,q} \tag{19}$$

7) MINIMUM DELAY CONSTRAINT

If the sum of end-to-end delays is limited to a minimum, there are more network resource to transmit other messages with low priorities. Meanwhile, it leaves an adequate safe distance between the arrived time and receipt time to keep the time-triggered message from missing receipt time in a destination end system. Moreover, the sum of end-to-end delays is limited to a minimum, so that there is enough time resource to select an appropriate time window for designing the receipt time. Time-triggered messages are not only high determinacy but also high real-time.

$$D = \sum_{m_p \in M} D_{m_p} = min \tag{20}$$

E. SMT-BASED SCHEDULING ALGORITHM

The basic constraint formulae are not specific to SMT solvers, but the task- and network-level schedule co-synthesis can be transformed into a Mixed Integer Programming (MIP) problem to minimize end-to-end delay [12]. All the switches send the TT frames according to the online TT schedule table dispatched by SDN centralized controller. To obtain an optimal online TT schedule table, the scheduling algorithm must be fast. However, it often costs too much time to finish resolving by MIP solver. For example, focused on a network with fewer than 64 TT messages, MIP solver Gurobi needs at least twenty times as the processing time as SMT solver Yices (the detailed values are about 50s/2s) [12]. So it is impractical to use MIP solver in the online scheduling context.

Algorithm 1 SMT-Based Scheduling Algorithm

```
1:  Data: $G(V, E), VL, M, F$;
2:  Result: $S(TT\ schedule\ table)$;
3:  Initialize: $delaymin \leftarrow \infty, delay \leftarrow 0, count \leftarrow 0, S \leftarrow \emptyset$;
4:  $Stmp \leftarrow SMTSolve(constaints)$;
5:  if $Stmp \neq \emptyset$ then
6:      while $Stmp \neq \emptyset || delaymin \neq 0.75delay$ do
7:          if $Stmp = \emptyset \&\& delaymin \neq 0.75delay$ then
8:              $delaymin \leftarrow 0.75delay$;
9:              update: $constaint(delaymin)(21)$;
10:         else
11:             $S \leftarrow SMTSolve(constaints)$;
12:             calculate: $delay$;
13:             $delaymin \leftarrow 0.5delay$;
14:             update: $constaint(delaymin)$ (21);
15:         end if
16:         $count{+}{+}$;
17:         $Stmp \leftarrow SMTSolve(constaints)$;
18:         if $time > timethd || count > forthd$ then
19:             $break$;
20:         end if
21:     end while
22: end if
23: return $S$;
```

In order to work with mixed time-triggered message periods, the algorithm need calculate the lowest common multiple (LCM) of all time-triggered message periods as a cluster cycle first. In the cluster cycle, the algorithm displays the number of each time-triggered message so that the time-triggered frames in all links can be known. According to the constraints 1-6, SMT can obtain a feasible solution for all frames. Finally, SMT-based scheduling algorithm with the constraints 1-7 can obtain a better solution by iterations.

The processing time of SMT solver Yices is feasible, but it can not support the optimal problem (e.g. the minimum). To solve the problem, the minimum delay constraint is converted as the following inequality, where the parameter delaymin is the predicted delay bound of all messages.

$$D = \sum_{m_p \in M} D_{m_p} < delaymin \qquad (21)$$

Though updating the parameter delaymin by dichotomy, the approximate optimal solution can be obtained by SMT solver Yices. Meanwhile, the 75% limitation is added into the scheduling algorithm to ensure that the minimum sum of end-to-end delays for all messages is not less than 75% of the calculated value, so as to guarantee the validity of the calculated value. Moreover, the thresholds of processing time and iteration are added to guarantee real-time online scheduling. Although the above algorithm can not obtain the optimal solution, but the SMT-based scheduling algorithm is a tradeoff between fast scheduling and optimal scheduling. If the SMT solver can not give a feasible solution, the traditional scheduling algorithm also has no feasible solution. The network can not load so many TT messages. Some TT messages should be degraded to rate-constrained messages. It does not happen under reasonable task requirements.

IV. ARCHITECTURE VALIDATION

A. EXPERIMENTAL STRATEGY

1) SIMULATION MODEL

The self-designed software based on OMNeT++ is used to perform experiments to evaluate the performance of the new method. The switch module and the controller module are similar to Klein and Jarschel [33] and Salih *et al.* [34]. But the buffers in the switch module and the centralized controller module are divided into three types, TT buffers, RC buffers, and BE buffers.

An ES module is displayed in Fig. 6, where Sflow modules generate all frames with different traffic types, Rflow modules receive all frames with different traffic types, flowDispatch module is responsible for dispatching frames to different Sflow/Rflow modules, buffer modules temporarily store different frames, and scheduler module realizes scheduling for different traffics according to traffic constraints discussed in section III-A. flowCheck module is used to check whether the received frames satisfy rules, such as the TT schedule table.

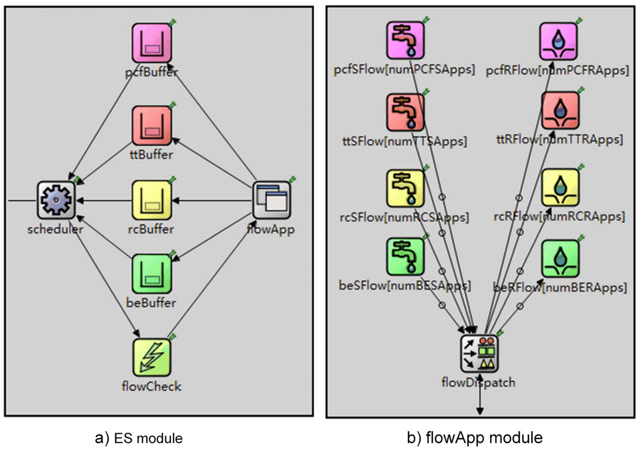


FIGURE 6. Implemented model of the ES.

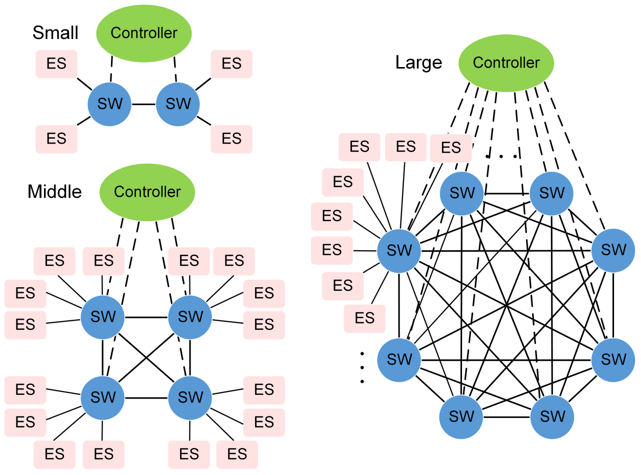


FIGURE 7. Network topology with three network sizes.

TABLE 1. Network configuration.

Scale	Quantities				
	switch	ES	TT	RC	BE
Small	2	4	16	60	52
Middle	4	16	32	300	180
Large	8	64	64	600	360

2) NETWORK CONFIGURATION

There are three types of network sizes for performance analysis in experiments. All networks are mesh network shown in Fig. 7, any two switches in the network are interconnected with each other. Meanwhile, there is a SDN centralized controller for each network to dispatch flow tables.

To verify the availability of SDTTE, a case with a small network is used. In middle and large networks, the number of messages grows to evaluate the performance of our algorithm. Detailed network configurations are shown in Table 1.

The bandwidth of each link is 100Mbit/s. When the period of TT messages is small (such as 2ms and 4ms), TT messages

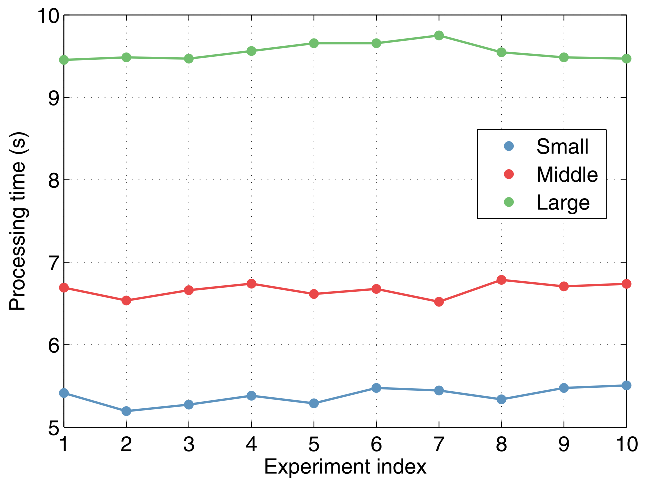


FIGURE 8. Processing time in different networks.

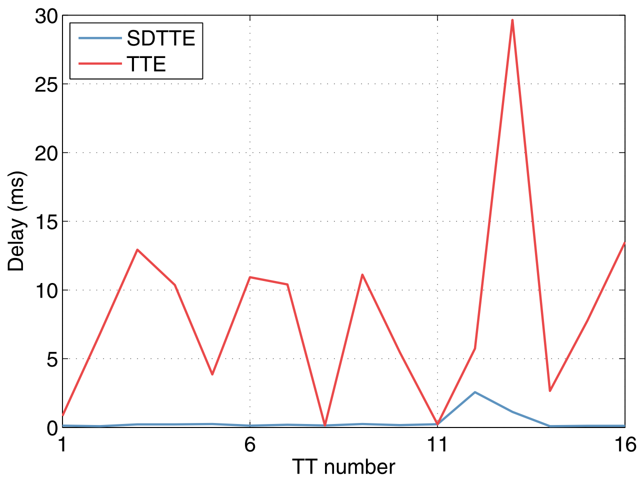


FIGURE 9. End to end delays for TT messages in a small mesh network.

can not be scheduled in the above networks as the solution space is too big to traverse. So in our experiment, the periods of TT messages are not same, they are randomly selected from the set (16ms, 32ms, and 64ms). The BAG of RC messages is a random value among 2ms, 4ms, and 8ms to make full use of bandwidth.

B. EXPERIMENT VERIFICATION

1) MECHANISM VERIFICATION

The SMT-based scheduling algorithm was processed by a laptop where operating system is Win7, CPU is i3-M380, memory is 4G, and software platform is VS2013 with VC++ language. The processing time of SMT-based scheduling algorithm is illustrated by ten independent repeated experiments for different networks as shown in Fig. 8, respectively. The results show that the processing time is less than 10s. The time drift is less than 1us during updating, when we assume that the clock drift is 100ns per second [35]. The threshold is mainly related to minimum gap between the generated time and the triggered time for a time-triggered message (it is 0.1ms in experiments). If the time drift becomes 0.1ms, it costs 1000s (about 15 minutes) at least. So the update frequency is more than 15 minutes. System and network are not synchronized, but system and network are synchronized respectively, such as SAE AS62580 in network synchronization. Thus the time drift between systems and network is much less than 100ns, the update frequency may be hour-level generally.

When the clock drift is bigger than 100ns per second or the update frequency is too high, it is feasible that compensation (some millisecond) can be added to the triggered time. Because the sum of end-to-end delays is limited to a minimum, there is an adequate safe distance between the arrived time and receipt time to keep the time-triggered message from missing receipt time in a destination end system. Moreover, jitters in Offset constraint can be changed to increase minimum gap between generated time and triggered time. The threshold can be higher by using the above two methods so that the update frequency becomes lower.

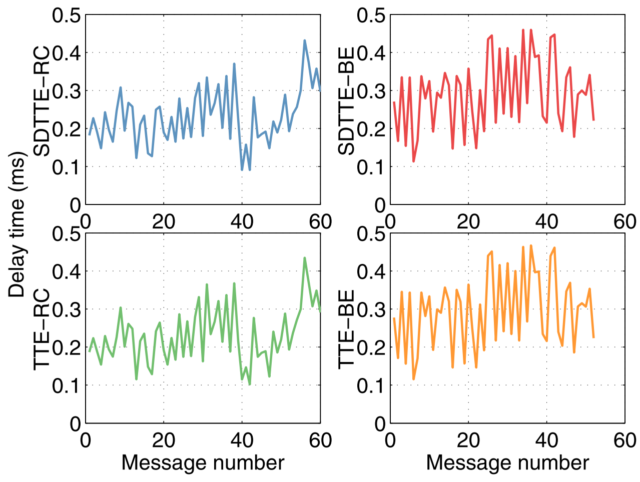


FIGURE 10. End to end delays for RC and BE messages in a small mesh network.

2) AVAILABILITY VERIFICATION

The small mesh network is used to verify the availability of SDTTE and the simulation results are shown in Fig. 9, which can illustrate the correctness of our simulation model. According to the results, the end to end delays for TT messages are almost reduced from the millisecond range to the microsecond range. Meanwhile, the end to end delays for RC messages and BE messages only have some slight changes which are shown in Fig. 10. It is verified that SDTTE can reduce the end to end delays for TT messages to guarantee their real-time and determinacy requirements. At the same time, SDTTE has no side effect on the RC traffic and the BE traffic.

C. EXPERIMENTAL RESULT AND ASSESSMENT

1) MIDDLE NETWORK

To obtain statistical regularity of end to end delay, the middle size network case is used to evaluate the performance of our model and algorithm, and the simulation results are shown in Fig. 11. It is clear that the end to end delays for TT

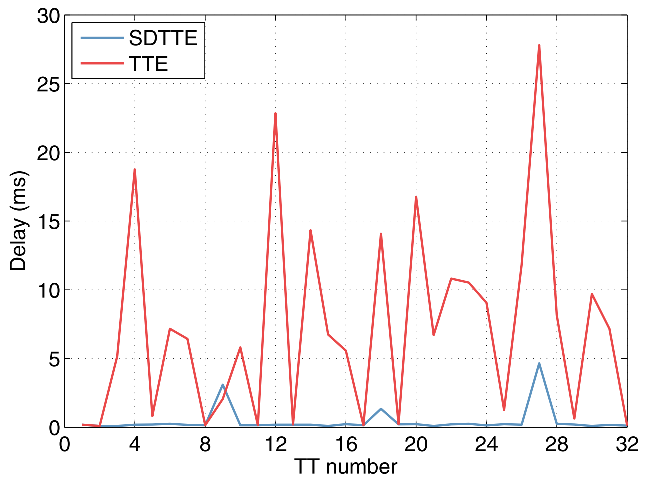


FIGURE 11. End to end delays for TT messages in a middle mesh network.

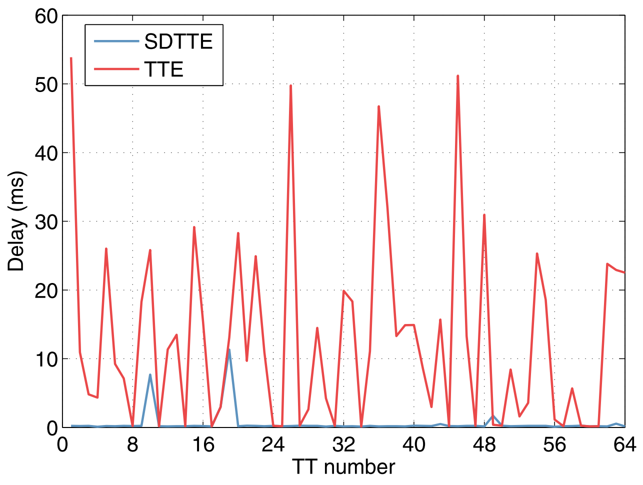


FIGURE 13. End to end delays for TT messages in a large mesh network.

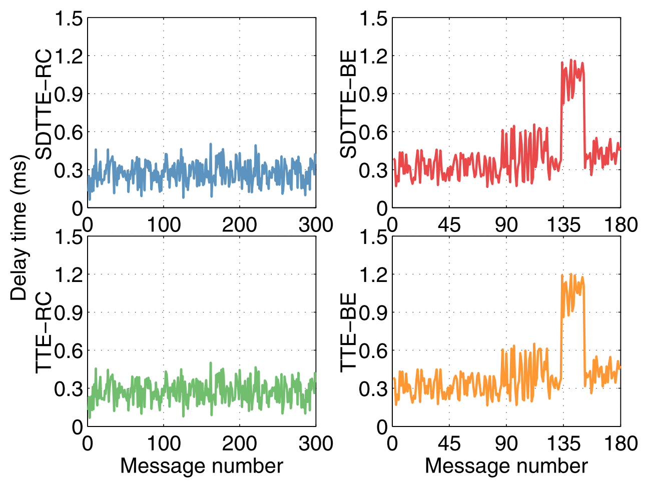


FIGURE 12. End to end delays for RC and BE messages in a middle mesh network.

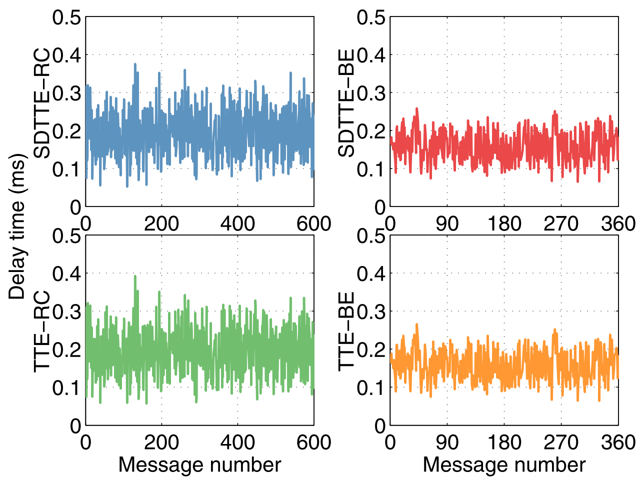


FIGURE 14. End to end delays for RC and BE messages in a large mesh network.

messages are also reduced from the millisecond range to the microsecond range, except the 9^{th}, 18^{th}, and 27^{th} TT messages. Meanwhile, the trends of the end to end delays for RC messages and BE messages are similar as in the small size network case, which are shown in Fig. 12. The delay differences between in TTE and SDTTE are quite small and the changing rate is less than 10%.

To minimize the sum of end to end delays for TT messages, it just means that probably not all the generated time of all TT messages can match with their triggered time. It depends on the detailed traffic configuration and topology. In some case, the difference between the generated time and the triggered time in SDTTE might become larger than that in TTE. It leads that the 9^{th} TT message in SDTTE has a bigger end to end delay. Under the constraint about minimizing the sum of end to end delays for TT messages, it is allowed that the delays for some TT messages could be large in order to obtain a holistic optimal solution. But the big delay is still smaller than the corresponding deadline which is 32ms. In another word, the calculated scheduling table still meets the real-time requirement for the 9^{th} TT message.

2) LARGE NETWORK

The end to end delays in a large mesh network are obtained in Fig. 13–14. The trends of the delay differences are the same as those in a middle mesh network. The results show that the end to end delays for TT messages in SDTTE are far less than those in TTE. Similarly, the end to end delays for RC and BE messages in SDTTE are approximate to those in TTE.

Just like in the middle size network, the generated time of some TT messages can not match well with their triggered time, such as the 10^{th} and 19^{th} TT messages. But still, their transmission deadlines are satisfied according to the results.

D. PHENOMENA ANALYSIS

The distribution of end to end delays for TT messages in the three networking cases are illustrated in Fig. 15. The result shows that end to end delays for TT messages in SDTTE

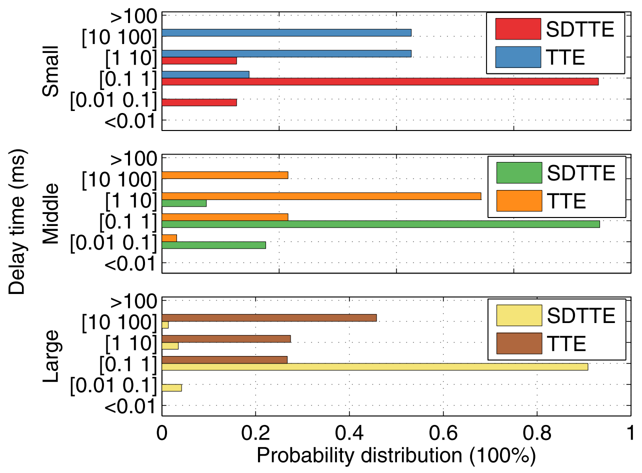


FIGURE 15. Statistical distribution of end to end delays for TT messages in mesh networks.

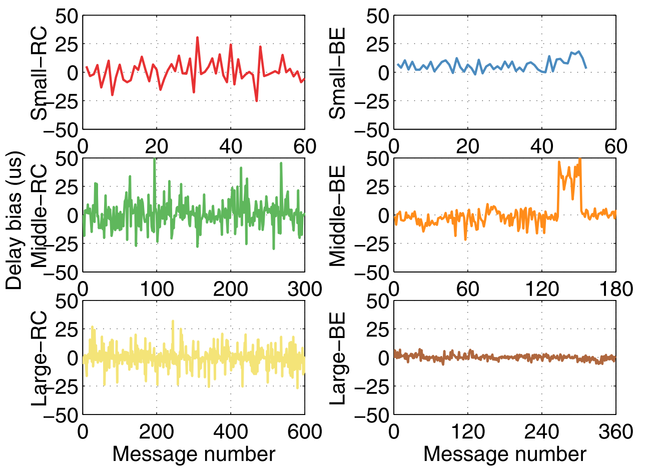


FIGURE 16. Delay biases for RC and BE messages in different mesh networks.

TABLE 2. End to end (e2e) delays in different scales of network.

Scale	Type	Mean e2e delay (us)			Median e2e delay (us)		
		TT	RC	BE	TT	RC	BE
Small	TTE	8273.1	231.0	297.3	7284.7	223.6	306.4
	SDTTE	372.8	230.4	291.2	171.6	225.7	291.8
Middle	TTE	7225.7	274.7	426.7	6560.0	278.2	366.7
	SDTTE	432.6	273.9	425.4	173.0	275.1	373.8
Large	TTE	12834.4	196.1	158.6	10301.0	195.6	156.5
	SDTTE	553.6	196.1	158.6	185.0	196.4	157.2

TABLE 3. Mean delay jitters in different scales of network.

Scale	Type	Mean delay jitter (us)		
		TT	RC	BE
Small	TTE	0.44	396.31	1170.06
	SDTTE	0.34	391.41	1112.39
Middle	TTE	0.49	552.47	1301.64
	SDTTE	0.34	552.38	1296.73
Large	TTE	0.38	347.82	622.02
	SDTTE	0.35	349.56	621.64

have a tighter distribution than that in TTE. The determinacy for TT messages in SDTTE is also improved. The most of end to end delays for TT messages are concentrated at the sub-millisecond level. Meanwhile, the distribution of end to end delays for TT messages is from an irregular distribution in TTE to a Poisson-like distribution in SDTTE as shown in Fig. 15. In some sense, It just agrees with the rule "The arrival of a message is a Poisson process" in queuing theory.

Statistical values about end to end delays in different scales of network are shown in Table 2. Compared with TTE, the mean end to end delays for TT messages in SDTTE are reduced to 4.5%, 5.9%, and 4.3% in the small, middle, and large network, respectively. The reducing rate in the middle network is less than those in other networks. The reason lies in that the 9^{th} TT message in SDTTE has a bigger delay than that in TTE, which just pulls up the reducing rate. If necessary, the separated constraints for each TT messages can be added in the last iteration process.

When there are some messages like the 9^{th} TT message in Fig. 10, the median end to end delays for TT messages can be used to analyze the overall performance. Compared with TTE, the median end to end delays for TT messages in SDTTE are reduced to 2.3%, 2.6%, and 1.8% in the small, middle, and large networks, respectively. The performance of TT messages in SDTTE is improved by about 40 times.

Meanwhile, Table 2 also shows that median end to end delays for TT messages are less than those of RC messages and BE messages, which means most of the TT messages can be optimized well.

Moreover, delay biases for RC and BE messages between SDTTE and TTE are also calculated and illustrated in Fig. 16. The results show that SDTTE has a sub-millisecond level impact on RC and BE messages. The mean delay biases for RC and BE messages in SDTTE are less than 1% of end to end delays for RC and BE messages. So the influence for RC and BE messages in SDTTE can be ignored.

Additionally, mean delay jitters for different types of messages are shown in Table 3, where the time synchronization accuracy is set as same as the time tick (20ns). The results show that the delay jitter for TT messages is the smallest, followed by the delay jitter for RC messages, and the delay jitter for BE messages is the largest. Combined with table 2 and table 3, it can be seen that SDTTE not only takes full advantages of strict deterministic transmission in the TT traffic, but also greatly reduces the end to end delay for the TT traffic in the task-level. When message configuration is reasonable, messages could have low delay and jitter so that they can obtain good real-time and determinacy.

V. CONCLUSION

The synchronization between tasks and the network is a critical prerequisite for the TT traffic to ensure real-time and determinacy requirement. Due to the asynchrony between tasks and the network, the end to end delays for TT messages are uncertain.

Detecting the generated time of TT messages is the critical mechanism to minimize the end to end delays. To make the generated time of TT messages detectable, the TT traffic is regarded as the event-triggered traffic with the highest priority in the source ES. The detailed generated time can be indirectly detected when TT messages arrive at the first connected switch.

And then SDN is introduced into TTE as SDTTE to detect the generated time of TT messages in different patterns and obtain the online TT schedule table. Thereafter, TT scheduling constraints is revised so that SMT can replace MIP to solve optimization problem. Meanwhile, SMT-based scheduling algorithm is optimized to process fast for online TT schedule.

Subsequently, the small scale case is used to verify the availability of SDTTE. Finally, two middle and large cases are realized to analyze the performance of SDTTE. The statistical results show that the end to end delays for TT messages are reduced by 95% and the end to end delays for RC and BE messages are almost unchanged. Meanwhile, the processing time is less than 10s, the threshold triggered the update avoids big jitters for time-triggered messages during updating.

ACKNOWLEDGMENT
The authors wish to acknowledge the efforts of all the entities contributing to the Open Networking Foundation (ONF) and TTTech. The colleagues who provided insight and stimulating discussions that greatly assisted in the preparation of this article. They also gratefully acknowledge the anonymous reviewers for carefully reading the article and providing constructive comments.

REFERENCES

[1] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer, "The time-triggered Ethernet (TTE) Design," in *Proc. 8th IEEE Int. Symp. Object-Oriented Real-Time Distrib. Comput. (ISORC)*, Seattle, WA, USA, Jun. 2005, pp. 22–33, doi: 10.1109/isorc.2005.56.

[2] W. Steiner, G. Bauer, B. Hall, M. Paulitsch, and S. Varadarajan, "TTEthernet dataflow concept," in *Proc. 8th IEEE Int. Symp. Netw. Comput. Appl.*, Cambridge, MA, USA, Jul. 2009, pp. 319–322, doi: 10.1109/nca.2009.28.

[3] R. Obermaisser, *Time-Triggered Communication*. Boca Raton, FL, USA: CRC Press, 2018.

[4] D. Tămaş-Selicean, P. Pop, and W. Steiner, "Design optimization of TTEthernet-based distributed real-time systems," *Real-Time Syst.*, vol. 51, no. 1, pp. 1–35, Jan. 2015, doi: 10.1007/s11241-014-9214-8.

[5] AEE Committee, "Arinc specification 664p7: Aircraft data network, part 7: Avionics full-duplex switched Ethernet (AFDX) network," Aeronautical Radio Inc, Annapolis, MD, USA, Tech. Rep., 2005.

[6] H. Kopetz, "Event-triggered versus time-triggered real-time systems," in *Operating Systems of the 90s and Beyond*. Berlin, Germany: Springer, 1991, pp. 86–101, doi: 10.1007/BFb0024530.

[7] S. Gopalakrishnan, M. Caccamo, and L. Sha, "Switch scheduling and network design for real-time systems," in *Proc. 12th IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS)*, San Jose, CA, USA, Apr. 2006, pp. 289–300, doi: 10.1109/rtas.2006.42.

[8] E. Suethanuwong, "Scheduling time-triggered traffic in TTEthernet systems," in *Proc. IEEE 17th Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Krakow, Poland, Sep. 2012, pp. 1–4, doi: 10.1109/etfa.2012.6489749.

[9] S. S. Craciunas, R. S. Oliver, and V. Ecker, "Optimal static scheduling of real-time tasks on distributed time-triggered networked systems," in *Proc. IEEE Emerg. Technol. Factory Autom. (ETFA)*, Barcelona, Spain, Sep. 2014, pp. 1–8, doi: 10.1109/etfa.2014.7005128.

[10] C. Barrett and C. Tinelli, "Satisfiability modulo theories," in *Handbook of Model Checking*. Cham, Switzerland: Springer, 2018, pp. 305–343.

[11] R. Sebastiani, "Lazy satisfiability modulo theories," *J. Satisfiability, Boolean Model. Comput.*, vol. 3, nos. 3–4, pp. 141–224, 2007.

[12] 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.

[13] C. Scholer, R. Krenz-Baath, A. Murshed, and R. Obermaisser, "Computing optimal communication schedules for time-triggered networks using an SMT solver," in *Proc. 11th IEEE Symp. Ind. Embedded Syst. (SIES)*, Krakow, Poland, May 2016, pp. 1–9, doi: 10.1109/sies.2016.7509415.

[14] 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, doi: 10.1109/rtss.2010.25.

[15] F. Pozo, W. Steiner, G. Rodriguez-Navas, and H. Hansson, "A decomposition approach for SMT-based schedule synthesis for time-triggered networks," in *Proc. IEEE 20th Conf. Emerg. Technol. Factory Autom. (ETFA)*, Luxembourg, Sep. 2015, pp. 1–8, doi: 10.1109/etfa.2015.7301436.

[16] D. Tamasselicean, P. Pop, and W. Steiner, "Timing analysis of rate constrained traffic for the TTEthernet communication protocol," in *Proc. IEEE 18th Int. Symp. Real-Time Distrib. Comput.*, Auckland, New Zealand, Apr. 2015, pp. 119–126, doi: 10.1109/isorc.2015.32.

[17] J. Yao, X. Xu, and X. Liu, "MixCPS: Mixed time/event-triggered architecture of cyber–physical systems," *Proc. IEEE*, vol. 104, no. 5, pp. 923–937, May 2016, doi: 10.1109/jproc.2016.2519381.

[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] M. Boyer, H. Daigmorte, N. Navet, and J. Migge, "Performance impact of the interactions between time-triggered and rate-constrained transmissions in TTEthernet," *Die Medizinische Welt*, vol. 14, no. 2, pp. 217–225, 2016, doi: 10.1017/S0021859600022838.

[20] J. Forget, F. Boniol, E. Grolleau, D. Lesens, and C. Pagetti, "Scheduling dependent periodic tasks without synchronization mechanisms," in *Proc. 16th IEEE Real-Time Embedded Technol. Appl. Symp.*, Stockholm, Sweden, Apr. 2010, pp. 301–310, doi: 10.1109/rtas.2010.26.

[21] W. Steiner, "Synthesis of static communication schedules for mixed-criticality systems," in *Proc. 14th IEEE Int. Symp. Object/Compon./Service-Oriented Real-Time Distrib. Comput. Workshops*, Mar. 2011, pp. 11–18, Newport Beach, CA, USA, doi: 10.1109/isorcw.2011.12.

[22] D. Tamas-Selicean, P. Pop, and W. Steiner, "Synthesis of communication schedules for TTEthernet-based mixed-criticality systems," in *Proc. 8th IEEE/ACM/IFIP Int. Conf. Hardw./Softw. Codesign Syst. Synth.*, Tampere, Finland, 2012, pp. 473–482, doi: 10.1145/2380445.2380518.

[23] L. Zhang, D. Goswami, R. Schneider, and S. Chakraborty, "Task-and network-level schedule co-synthesis of Ethernet-based time-triggered systems," in *Proc. 19th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Singapore, Jan. 2014, pp. 119–124, doi: 10.1109/aspdac.2014.6742876.

[24] 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.

[25] K. Sampigethaya, "Software-defined networking in aviation: Opportunities and challenges," in *Proc. Integr. Commun., Navigat. Surveill. Conf. (ICNS)*, Herdon, VA, USA, Apr. 2015, pp. 1–21, doi: 10.1109/icnsurv.2015.7121310.

[26] P. Heise, F. Geyer, and R. Obermaisser, "Deterministic openflow: Performance evaluation of SDN hardware for avionic networks," in *Proc. 11th Int. Conf. Netw. Service Manage. (CNSM)*, Barcelona, Spain, Nov. 2015, pp. 372–377, doi: 10.1109/cnsm.2015.7367385.

[27] J. W. Guck, A. Van Bemten, and W. Kellerer, "DetServ: Network models for real-time QoS provisioning in SDN-based industrial environments," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 4, pp. 1003–1017, Dec. 2017, doi: 10.1109/tnsm.2017.2755769.

[28] R. Kumar, M. Hasan, S. Padhy, K. Evchenko, L. Piramanayagam, S. Mohan, and R. B. Bobba, "End-to-end network delay guarantees for real-time systems using SDN," in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, Paris, France, Dec. 2017, pp. 231–242, doi: 10.1109/rtss.2017.00029.

[29] S. S. Craciunas and R. S. Oliver, "SMT-based task-and network-level static schedule generation for time-triggered networked systems," in *Proc. 22nd Int. Conf. Real-Time Netw. Syst.-RTNS*, Versailles, France, 2014, p. 45, doi: 10.1145/2659787.2659812.

[30] J. Grieu, "Analyse et évaluation de techniques de commutation Ethernet pour l'interconnexion des systèmes avioniques," Ph.D. dissertation, Inst. Nat. Polytechnique de Toulouse, Labège, France, 2004.

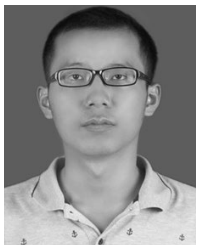
[31] C. Fouard, G. Malandain, S. Prohaska, and M. Westerhoff, "Blockwise processing applied to brain microvascular network study," *IEEE Trans. Med. Imag.*, vol. 25, no. 10, pp. 1319–1328, Oct. 2006, doi: 10.1109/TMI.2006.880670.

[32] M. Boyer, J. Migge, and M. Fumey, "PEGASE-a robust and efficient tool for worst-case network traversal time evaluation on AFDX," *Physiol. Behav.*, vol. 25, no. 4, pp. 589–593, 2011, doi: 10.4271/2011-01-2711.

[33] D. Klein and M. Jarschel, "An openflow extension for the OMNeT++ inet framework," in *Proc. 6th Int. ICST Conf. Simulation Tools Techn.*, Cannes, France, 2013, pp. 322–329, doi: 10.4108/simutools.2013.251722.

[34] M. A. Salih, J. Cosmas, and Y. Zhang, "OpenFlow 1.3 extension for OMNeT++," in *Proc. IEEE Int. Conf. Comput. Inf. Technol.; Ubiquitous Comput. Commun.; Dependable, Autonomic Secure Comput.; Pervasive Intell. Comput.*, Liverpool, U.K., Oct. 2015, pp. 1632–1637, doi: 10.1109/cit/iucc/dasc/picom.2015.246.

[35] J. Zhao, Q. Yuan, H. Sun, J. Yang, and L. Sun, "A high-performance oscillator based on RF MEMS resonator and low-noise sustaining circuit for timing applications," *IEICE Electron. Express*, vol. 15, no. 10, 2018, Art. no. 20180395.



JUN LU was born in Jingzhou, Hubei, China. He received the M.S. degree in control engineering from the School of Electronics and Information, Northwestern Polytechnical University, Xi'an, China, in September 2014. He is currently pursuing the Ph.D. degree in communication and information system with the School of Electronic Information Engineering, Beihang University, China.

His research interests include avionics information integration, software defined networks, and embedded systems.

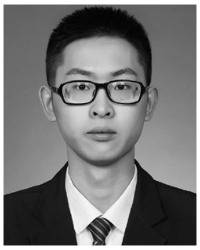
HUAGANG XIONG received the Ph.D. degree in communication and information system from the School of Electronic Information Engineering, Beihang University, China, in 1998.

He is currently a Full Professor with Beihang University. He has published more than 305 peer-reviewed articles which are indexed by SCI or EI and three books. He has presided more than 20 major projects in total, such as the National Natural Science Foundation of China, national 863 program, and a Civil Aircraft Research. His research interests include communication network theory and technology, avionics information integration, airborne network, and standards.

Dr. Xiong is a member of the China Aviation Electronics Standardization Committee and the Avionics and Air Traffic Control Branch of China Society of Aeronautics and Astronautics. He is the Director of the Beijing Electronic Circuit Research Association and an Expert of the Civil Aircraft Scientific Research Group. He is the Chief of the BUAA-TTTech Time-Triggered Technology Joint Laboratory (TTTJL), Beihang University, where he is also the Head of the Avionics and Bus Communications Research Team (ABC), School of Electronic Information Engineering.

FENG HE received the Ph.D. degree in communication and information systems from the School of Electronic Information Engineering, Beihang University, China, in 2008.

He is currently an Associate Professor with the School of Electronic Information Engineering, Beihang University. He has published more than 76 peer-reviewed articles and two books. He has presided more than ten major projects in total, such as the National Natural Science Foundation of China, national 863 program, and a Civil Aircraft Research. His research interests concern digital communication technology, communication network theory and technology, avionics integration, software defined networks, embedded systems, and real-time networks.



RONGWEI WANG received the B.S. degree in electronic and information engineering from Harbin Engineering University, Harbin, China, in 2017. He is currently pursuing the M.S. degree with the School of Electronic Information Engineering, Beihang University, China. His main research directions are avionics information integration and software defined networks.