2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS) | 978-1-6654-7177-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICDCS54860.2022.00072

E-TSN: Enabling Event-triggered Critical Traffic in Time-Sensitive Networking for Industrial Applications

Yi Zhao*, Zheng Yang*‡, Xiaowu He*, Jiahang Wu*, Hao Cao*, Liang Dong*, Fan Dang†, and Yunhao Liu†
* School of Software and BNRist, Tsinghua University † Global Innovation Exchange, Tsinghua University
‡ Corresponding author
{zhaoyi.yuan31, hmilyyz, horacehxw, jiahangok, i.haocao}@gmail.com, thdongl@163.com,
dangfan@tsinghua.edu.cn, yunhao@greenorbs.com

***Abstract*—Time-Sensitive Networking (TSN) is the most promising network technology for Industry 4.0. A series of IEEE standards on TSN introduce deterministic transmission into standard Ethernet. Under the current paradigm, TSN can only schedule the deterministic transmission of time-triggered critical traffic (TCT), neglecting the other type of traffic in industrial cyber physical systems, *i.e.*, event-triggered critical traffic (ECT). So in this work, we propose a new paradigm for TSN scheduling named E-TSN, which can provide deterministic transmission for both TCT and ECT. The three techniques of E-TSN, *i.e.*, probabilistic stream, prioritized slot sharing, and prudent reservation, enable the deterministic transmission of ECT in TSN, and at the same time, protect TCT from the impacts of ECT. We also develop and make public a TSN evaluation toolkit to fill the gap in TSN study between algorithm design and experimental validation. The experiments show that E-TSN can reduce the latency and jitter of ECT by at least an order of magnitude compared to state-of-the-art methods. By enabling reliable and timely delivery of ECT in TSN for the first time, E-TSN can broaden the application scope of TSN in industry.**

***Index Terms*—Time-Sensitive Networking, Event-triggered critical traffic, Traffic Scheduling, Cyber physical system.**

I. Introduction

Time-Sensitive Networking (TSN) is considered the most important technology for future industrial communications by industry leaders [1]–[3]. It empowers Industry 4.0 by uniting diverse parts of an enterprise, including Information Technology (IT) and Operation Technology (OT) sectors.

TSN is an OSI layer 2 technology, which provides deterministic communication on standard Ethernet. It ensures that messages can travel through the network in a fixed and predictable amount of time. The switches and devices in a TSN network are time-synchronized. A global schedule controls when data streams are generated and transmitted by devices and switches. As a result, the switches can reserve resources in advance to guarantee the deterministic transmission of data streams. Such traffic that happens periodically at the predetermined time is called time-triggered traffic in industrial communication, such as a sensor reporting pressure every 10ms.

The other main type of traffic in industrial cyber physical systems is event-triggered traffic. Event-triggered traffic is initiated by the happening of some event [4]. The fundamental difference between time- and event-triggered traffic is that the occurrence time of the latter is uncertain. Both time-triggered critical traffic (TCT) and event-triggered critical traffic (ECT) are common in reality. For example, Tunnel Boring Machine (TBM) is advanced equipment for tunneling, such as subway tunnels. Fig. 1 shows a TBM and its operator cabin. The operator closely monitors the status of TBM and gives proper commands to TBM by pushing the buttons in an emergency. The operator's commands and some monitored statuses, such as cutterhead hazard, are typical ECT data. To ensure that the operator's commands are immediately and reliably executed by the TBM, the operation panel is directly connected to controlled devices electrically or mechanically. The inability to deterministically transmit such ECT through networks prevents the digitalization of TBM, *e.g.*, automatically altering operation parameters to increase cutter life. It also forces the operator to stay in the tunnel with high temperature and humidity, and most importantly, security issues. Besides TBM, ECT is also seen frequently in manufacturing and automotive systems [4]–[7].

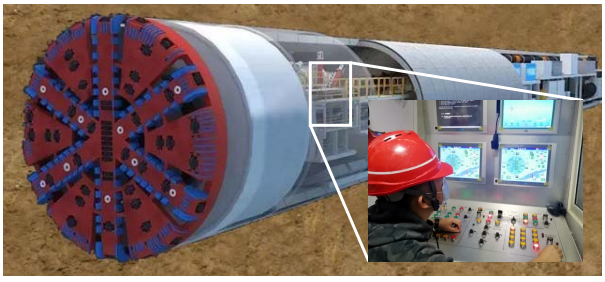


Fig. 1: TBM and its operator cabin. The signal of the emergency button is typical event-triggered critical traffic.

Under the current paradigm, TSN can only support the deterministic transmission of TCT. The scheduling problem of TCT has been studied by much previous work [8]–[11]. However, there is a gap in the research of ECT in TSN. Enabling

2575-8411/22/$31.00 ©2022 IEEE
DOI 10.1109/ICDCS54860.2022.00072

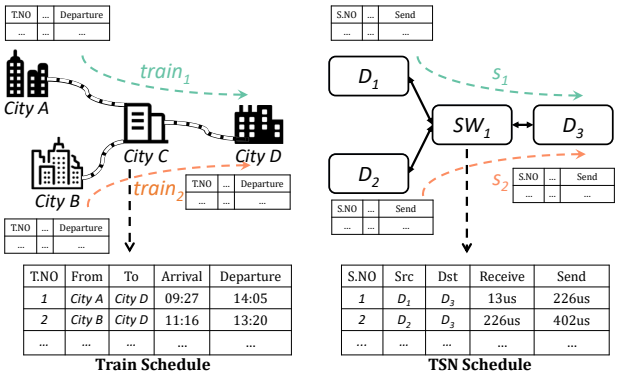


Fig. 2: The schedule of a train system and the schedule of a TSN network. Three devices (D) and one switch (SW) form a TSN network on the right. The train system on the left is an analogy to the TSN network on the right. A TSN schedule specifies the time when traffic passes each link in the network, just like a train schedule specifies the time when trains pass each railway.

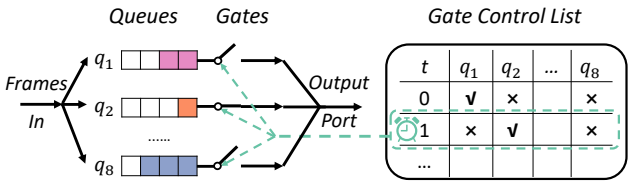


Fig. 3: Output port model of TSN switch. The frames are dispatched to different queues in the switch. Then the frames in different queues are selected for transmission based on current time and the Gate Control List. ✓means the queue can be selected for transmission and × means the queue cannot be selected.

deterministic transmission of ECT in TSN is challenging in three ways: (1) **Unpredictable traffic**. The occurrence of events is random and unpredictable. As a result, existing TSN traffic scheduling based on predetermined timetables cannot deal with ECT. It is critical to model the different possibilities of ECT for scheduling. (2) **Unguaranteed delay**. If we reserve fixed time-slots for ECT, like what previous work does for TCT, the ECT needs to wait for the entire interval between two allocated time-slots in the worst case. (3) **Unprotected encroachment**. If we allow ECT to encroach upon TCT's resources without protection, the QoS of TCT will degrade unboundedly.

In this paper, we propose a new paradigm for TSN scheduling named E-TSN, which can provide deterministic transmission for both TCT and ECT. Targeting the three challenges mentioned above, E-TSN has three novel techniques that differentiate it from the traditional paradigm of TSN scheduling: (1) Probabilistic stream. We propose probabilistic stream to model the different possibilities of ECT, and we allow a time-slot to be in "superposition state" when frames of different possibilities are scheduled at the same time-slot. (2) Prioritized slot sharing. We let ECT share the time-slots of TCT, so ECT can be immediately transmitted whenever it occurs. (3) Prudent reservation. We propose a link-level time-slot reservation algorithm and model the worst-case latency of TCT to ensure that its requirements are not violated. We formalize the above techniques and the complete joint scheduling algorithm as a Satisfiability Modulo Theories [12] (SMT) problem, and we implement a TSN testbed on FPGA to evaluate the performance of E-TSN. The contributions of our work are summarized as follows:

- To the best of our knowledge, this is the first work to discuss and solve the problem of ECT's reliable and timely delivery in TSN. E-TSN also complies with IEEE TSN standards. Therefore it can operate in off-the-shelf TSN switches.
- E-TSN is a new paradigm for TSN scheduling based on three novel techniques: Probabilistic stream enables the modeling of unpredictable ECT. Prioritized slot sharing guarantees the low latency of ECT whenever it happens. Prudent reservation protects TCT from the impacts of ECT.
- We develop TSN platform Ziggo[1] from scratch to evaluate the effectiveness of our method. The platform aims to fill the gap in TSN study between algorithm design and experimental validation. It supports major TSN standards including 802.1AS [13], Qav [14], Qbv [15], and Qcc [16].
- We evaluate the performance of E-TSN in both testbeds and simulations. The results show that E-TSN enables ECT in TSN by providing an order of magnitude lower latency (423μs over three hops) and jitter (39μs standard deviation) compared to state-of-the-art methods.

The remaining of the paper is organized as follows. First, we discuss the preliminaries of TSN in Sec. II. Then we present the overview and three techniques of E-TSN in Sec. III. The complete formalization of the scheduling problem is in Sec. IV. We introduce the implementation of the testbed in Sec. V and evaluate the performance of E-TSN in Sec. VI. At last, we discuss the related work in Sec. VII and conclude the paper in Sec. VIII.

II. Preliminary

In this section, we introduce the preliminaries of TSN and its scheduling. In Fig. 2, we use a train system as an analogy to explain what TSN scheduling does. On the left, a simple train system has four cities connected by three railways. Two trains travel through this railway network. One train travels from City A to D, and the other travels from City B to D. They both go through the railway between City C and D. To avoid conflicts, the train system manager should schedule the arrival and departure time at stations properly. This is similar to what TSN does to data streams. On the right of Fig. 2, three

[1]TSN platform Ziggo: http://tns.thss.tsinghua.edu.cn/ziggo/

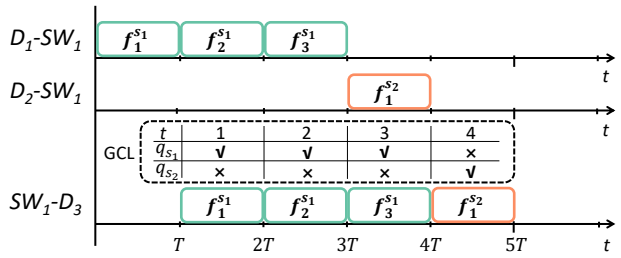


Fig. 4: An example schedule for the scheduling problem in Sec. II. $f_j^{s_i}$ is the j^{th} frame of s_i sent in a period. T is the time to transmit one frame on a link. GCL can be derived from the schedule. q_{s_i} is the queue for s_i.

devices and one switch form a simple TSN network. Since at one time, only one stream can transmit on a physical link. The sending and receiving time of streams at switches also need to be scheduled in advance to avoid conflicts and ensure determinism.

The ability to transmit frames according to predetermined timetable comes from standard 802.1Qbv [15]. As shown in Fig. 3, an output port of an 802.1Q [17] switch can have at most eight queues for frames of different priorities. The amendment Qbv further defines a gate to each queue that controls whether the frames in it can be transmitted. The status of the gate, *i.e.*, open or close, is predetermined in a timetable called Gate Control List (GCL). The entries in a GCL specify the status of each queue's gate in each time-slot in a cycle. Therefore, when some frames are scheduled to send in a time-slot, the gate of these frames' queue will be set to open, and the gates of the other queues will be set to close.

At last, we explain in more detail how TSN scheduling works from the link-time perspective as shown in Fig. 4. Assume stream s_1 and s_2 on the right of Fig. 2 are two TCT streams. s_1 is from D_1 to D_3 and s_2 is from D_2 to D_3. s_1 sends three frames in one cycle and s_2 sends one. The cycle time is both $5T$, where T is the time to transmit one frame. And the maximum allowed latency of two streams is also $5T$. Fig. 4 shows a schedule of these two streams on three links. From t=$3T$ to $4T$, the frame of s_2 is transmitted from D_2 to SW_1. Then from t=$4T$ to $5T$, the frame is transmitted from SW_1 to D_3. Thus the latency of s_2 is $2T$, under its maximum allowed latency. Here we use a simple example to explain how scheduling works. In a large network, the scheduling problem can be rather complex.

III. Design of E-TSN

A. Overview

Fig. 5 shows the overview of E-TSN and its position in the framework of TSN. 802.1Qcc [16] defines four main components in a TSN network, including end devices, switches, Centralized User Configuration (CUC), and Centralized Network Configuration (CNC). CUC discovers the end devices in the network, retrieves their stream requirements, and configures their TSN features such as the sending time of data. CNC is

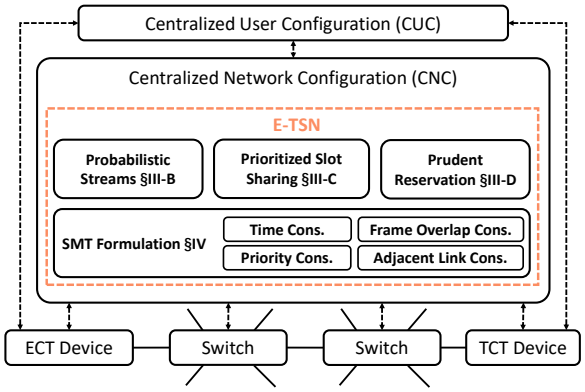


Fig. 5: Overview of E-TSN and its position in the framework of TSN.

aware of the physical topology of the network and receives stream requirements from CUC. E-TSN works inside CNC to calculate a schedule for the TSN network. Then CNC distributes this schedule to switches and end devices. Inside E-TSN, we propose three novel techniques, *i.e.*, probabilistic stream, prioritized slot sharing, and prudent reservation. These techniques and the complete scheduling problem are formalized as an SMT problem. The formalization is divided into four types of constraints, namely time constraints, frame overlap constraints, priority constraints, and adjacent link constraints.

E-TSN is fully compatible with current TSN standards and frameworks. ECT can be described by the user/network configuration information defined in Qcc 46.2. The three novel techniques to enable ECT in TSN can be implemented by configuring the GCL defined in Qbv.

In the rest of this section, we will present the three techniques of E-TSN.

B. Probabilistic Stream

The occurrence time of TCT is deterministic, which is predetermined by the scheduling algorithm. However, the occurrence of ECT is stochastic. It may arrive at any time when the system runs. This is their fundamental difference and makes it impossible for previous scheduling algorithms to model ECT.

In E-TSN, we propose the concept of probabilistic stream to model the different possibilities of ECT. We explain the idea of probabilistic stream using the network example on the right of Fig. 2. Now we assume that s_2 becomes an ECT stream, instead of TCT. The minimum time between consecutive events of s_2 is $5T$. This is a common property of ECT [5]. The period and maximum allowed latency of s_1 are $5T$. The maximum allowed latency of s_2 is also $5T$. s_1 sends three frames in one period, and s_2 sends one frame at a time. Stream s_2 can start to transmit at any time at D_2. We create N probabilistic streams, ps_{21}, ps_{22}, ..., ps_{2N} to represent the different possibilities of stream s_2. N is a parameter specified by the user. ps_{2i} is a time-triggered periodic stream that starts

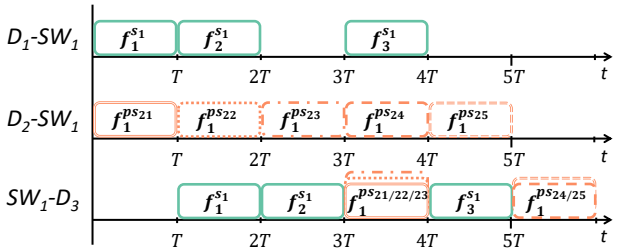


Fig. 6: An example schedule for the scheduling problem in Sec. III-B. $f_j^{s_1}$ represents the j^{th} frame of s_1. ps_{2i} represents the probabilistic streams of s_2. $f_j^{ps_{2i}}$ represents the j^{th} frame of ps_{2i}. T is the time to transmit one frame on a link.

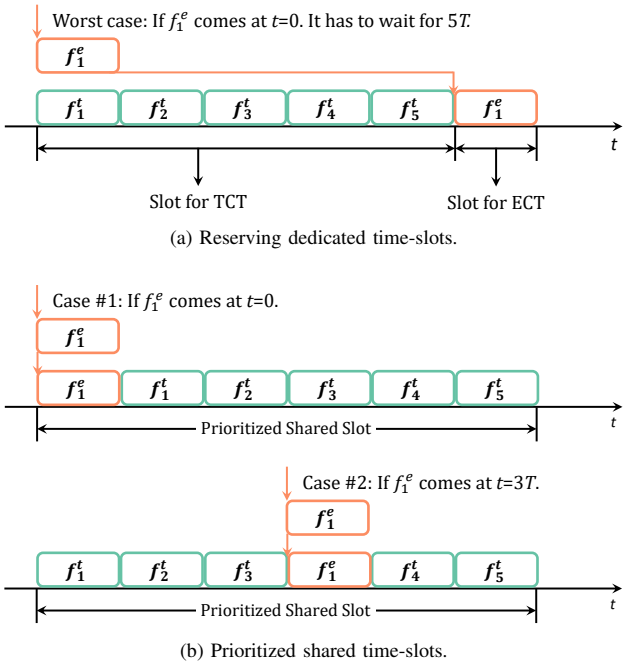
to transmit at $(i-1)(5T/N)$. It has the same source and destination as s_2. Its period is set to $5T$, *i.e.*, the minimum interevent time of the stream. If s_2 happens between ps_{i-1} and ps_i, it can be delayed for at most $5T/N$ to become ps_i. So its maximum allowed latency is set to $5T$-$5T/N$. Therefore, the set of probabilistic streams can represent all the possibilities of the ECT stream. If a schedule satisfies the latency requirements of all the probabilistic streams, it will satisfy the requirements of the ECT no matter when it happens.

In traditional TSN scheduling, two frames transmitted on the same link cannot overlap in time since a link can only transmit one frame at a time. For example, in Fig. 4, scheduling $f_1^{s_2}$ and $f_3^{s_1}$ to transmit at the same time on link SW_1-D_3 is invalid. This constraint is rooted in the determinism of TCT. However, this is not true after we introduce probabilistic streams. At one time, at most one of ps_{21}, ps_{22}, ..., ps_{2N} can actually happen. So we rethink this fundamental constraint in our scheduling algorithm. If two frames belong to different probabilistic streams derived from the same ECT stream, we allow them to overlap in time.

Fig. 6 illustrates an example schedule for TCT s_1 and ECT s_2. We use five probabilistic streams for s_2: ps_{21}, ps_{22}, ..., ps_{25}. A time slot on link SW_1-D_3 between $f_2^{s_1}$ and $f_3^{s_1}$ is reserved for 3 overlapped frames: $f_1^{ps_{21}}$, $f_1^{ps_{22}}$ and $f_1^{ps_{23}}$. It ensures that the latency requirements of ps_{21}, ps_{22} and ps_{23} are fulfilled. $f_1^{ps_{24}}$ and $f_1^{ps_{25}}$ are scheduled at the same time slot after $f_3^{s_1}$. This satisfies the requirements of ps_{24} and ps_{25}. As a result, all possibilities of s_2 will meet their transmission deadline. The above two time-slots are in "superposition state". The first one may transmit $f_1^{ps_{21}}$ or $f_1^{ps_{22}}$ or $f_1^{ps_{23}}$. There are multiple possibilities during scheduling, but only one of them can happen when the network is in operation.

C. Prioritized Slot Sharing

Probabilistic stream enables us to model ECT in the scheduling algorithm. However, reserving dedicated time-slots for ECT is inefficient. For example, Fig. 7a shows a schedule for a link. In this schedule, a slot is reserved exclusively for ECT every $6T$ period, and the other five slots are reserved for a TCT stream. In the worst case, when the event-triggered frame (f_1^e) appears just after its reserved slot, it has to wait for



(a) Reserving dedicated time-slots.

(b) Prioritized shared time-slots.

Fig. 7: Example schedule on a link with five TCT frames (f_i^t) and one ECT frame (f_1^e).

an entire period of $5T$ to be transmitted. This is unacceptable for critical traffic that needs low latency, and it also increases the jitter of ECT.

To tackle this problem, we propose prioritized slot sharing. ECT can transmit with a higher priority in the time-slots reserved for TCT. As shown in Fig. 7b, 5+1 time-slots are reserved for TCT. These time-slots are shared with ECT. So whenever f_1^e comes, it can be transmitted immediately. The remaining frames of TCT are delayed accordingly. To ensure that the delayed frames are also transmitted, we need to reserve some extra time-slots for TCT, such as the +1 time-slot in this example. As a result, with prioritized slot sharing, we can ensure the fast transmission of ECT whenever it happens. This significantly reduces the network latency and jitter.

D. Prudent Reservation

In prioritized slot sharing, we reserve extra shared time-slots rather than dedicated time-slots for ECT. To protect TCT from the encroachment of ECT on shared time-slots, we need to reserve extra time-slots for TCT and ensure that TCT's worst-case latency will not exceed the limit. Previous scheduling algorithms take it as a fundamental assumption that the length (number of frames) of the stream is constant along its path. However, reserving extra time-slots at the stream level can cause resource waste. Taking the network in Fig. 2 as an example again, assume that TCT stream s_1 shares its time-slots with ECT stream s_2. Reserving extra time-slots along the entire path of s_1 is unnecessary since s_1 and s_2 only overlap on link SW_1-D_3. To minimize the extra reserved time-slots,

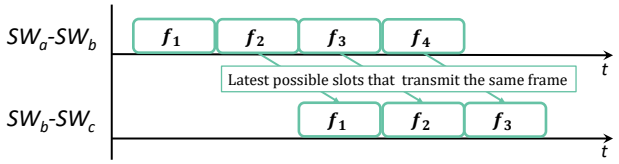


Fig. 8: Schedule the time-slots on SW_b-SW_c after the latest time-slot on SW_a-SW_b that may transmit the same frame.

we design a prudent reservation algorithm at link level. The algorithm is shown in Alg. 1. The inputs are the set of shared TCT streams and the set of ECT streams, and the outputs are the expanded TCT streams. For every link along a stream's path, we find all the ECT streams that pass this link and calculate the number of needed extra time-slots based on the length ($s_e.l$, $s_t.l$) and minimum interevent time ($s_e.T$) of the streams. T is the time to transmit one frame.

Algorithm 1: PRUDENTSLOTRESERVATION

Input: Set of Shared TCT streams S_1
Set of ECT streams S_2
Output: Expanded S_1^{exp} with extra frames added
forall s_t *of* S_1 **do**
  forall $link$ *of* $s_t.path$ **do**
    forall s_e *of* S_2 **do**
      if s_e *passes* $link$ **then**
        $n \leftarrow s_e.l \times \lceil s_t.l \times T / s_e.T \rceil$;
        Reserve n extra frames on $link$ for s_t;
return S_1

As a result of prudent slot reservation, the number of scheduled frames on adjacent links can be different. A frame's reserved time-slot on the downstream link should be after its time-slot on the upstream link. Due to the uncertainties of ECT, the correspondence of time-slots on adjacent links is indefinite. To ensure that a TCT frame won't miss its time-slot, we need to schedule the time-slots on the downstream link after the latest time-slot on the upstream link that may transmit the same frame. An example is shown in Fig. 8, the stream overlaps with an ECT stream on link SW_a-SW_b, but not on the downstream link SW_b-SW_c. So the stream reserves one more time-slot on link SW_a-SW_b. Then the time-slot of f_i on the downstream link should be scheduled after f_{i+1} on the upstream link, instead of f_i. After applying this rule along the entire path of the stream, we can model its worst-case latency as the time between the receiving on the last link and the sending on the first link.

IV. SMT FORMULATION

A. Network and Traffic Notation

We abstract the network topology as a directed graph $G(V, E)$, where graph vertices (V) represent switches and devices, and graph edges ($E \subseteq V \times V$) represent the links between them. If two network nodes v_a and v_b are connected, two edges, $\langle v_a, v_b \rangle$ and $\langle v_b, v_a \rangle$, will be added into E to represent the full-duplex link between them. An edge has three attributes:

$$(\langle v_a, v_b \rangle .b, \langle v_a, v_b \rangle .d, \langle v_a, v_b \rangle .tu)$$

b is the bandwidth of the link. d is the propagation delay. tu is the smallest time unit for the operations on the link. It determines the time granularity of the scheduling.

Next, we introduce the notation of streams. Without loss of generality, we only consider unicast streams to simplify the descriptions [8], [18]. A stream s can be characterized by 8 attributes:

$$(s.path, s.e2e, s.p, s.l, s.T, s.type, s.share, s.ot)$$

$s.path$=$[\langle v_1, v_2 \rangle, \langle v_2, v_3 \rangle ..., \langle v_{n-1}, v_n \rangle]$ is the path of the stream through the networks. $s.e2e$, $s.p$, and $s.l$ denote the maximum allowed end-to-end latency, priority, and length in bytes of the stream respectively. For TCT streams, $s.T$ is the period. For probabilistic streams derived from ECT streams, $s.T$ is the minimum interevent time. $s.type$ is the type of the stream: *Deterministic(Det)* for TCT streams or *Probabilistic(Prob)* for probabilistic streams. $s.share$ is only valid for TCT streams and indicates whether s shares its time-slots with ECT. $s.ot$ is only valid for probabilistic streams, which is its occurrence time, *i.e.*, when it starts to transmit at the source device. We denote the set of streams to schedule as S=$\langle s_1, s_2, s_3, ..., s_N \rangle$, including both TCT streams and the probabilistic streams derived from ECT.

At last, we introduce the notation of frames. We denote the set of frames on link $\langle v_a, v_b \rangle$ that belong to stream s_i as

$$F_{s_i,\langle v_a,v_b \rangle} = \left[f_1^{s_i,\langle v_a,v_b \rangle}, f_2^{s_i,\langle v_a,v_b \rangle}, ..., f_{last}^{s_i,\langle v_a,v_b \rangle} \right]$$

$F_{s_i,\langle v_a,v_b \rangle}$ includes the frames added by Alg. 1. Each frame has 3 attributes:

$$\left\langle f_j^{s_i,\langle v_a,v_b \rangle}.\phi, f_j^{s_i,\langle v_a,v_b \rangle}.T, f_j^{s_i,\langle v_a,v_b \rangle}.L \right\rangle$$

ϕ is the start time of the scheduled time slot for the frame. T is the period or minimum interevent time. L is the time to transmit this frame on link $\langle v_a, v_b \rangle$. All of them are in units of $\langle v_a, v_b \rangle .tu$.

B. Formulation

Based on the notation defined above, we present the complete SMT formulation of our scheduling algorithm, including how we implement the three techniques in Sec. III.

1) Time Constraints: First of all, the scheduled time for frames cannot have negative values, and the transmission of the frames should fit in their periods. Thus we have:

$$\begin{aligned} &\forall s_i \in S, \forall \langle v_a, v_b \rangle \in s_i.path, \forall f_j^{s_i,\langle v_a,v_b \rangle} \in F_{s_i,\langle v_a,v_b \rangle} : \\ &\quad \left(f_j^{s_i,\langle v_a,v_b \rangle}.\phi \geqslant 0 \right) \wedge \\ &\quad \left(f_j^{s_i,\langle v_a,v_b \rangle}.\phi + f_j^{s_i,\langle v_a,v_b \rangle}.L \leqslant f_j^{s_i,\langle v_a,v_b \rangle}.T \right) \end{aligned} \quad (1)$$

Specially, for a probabilistic stream, the scheduled ϕ for its first frame on the first link should be after its occurrence time:

$$\begin{aligned} &\forall s_i \in S \text{ if } s_i.type = Prob: \\ &\quad f_1^{s_i, s_i.path[0]}.\phi \geqslant s_i.ot \end{aligned} \tag{2}$$

where "if" means adding constraints to SMT conditionally.

Secondly, the frames of the same stream should be sent in sequence through a link:

$$\begin{aligned} &\forall s_i \in S, \forall \langle v_a, v_b \rangle \in s_i.path, \forall f_j^{s_i,\langle v_a,v_b\rangle} \in F_{s_i,\langle v_a,v_b\rangle}: \\ &\quad \text{if } f_j^{s_i,\langle v_a,v_b\rangle} \text{ is not } f_{last}^{s_i,\langle v_a,v_b\rangle}: \\ &\qquad f_j^{s_i,\langle v_a,v_b\rangle}.\phi + f_j^{s_i,\langle v_a,v_b\rangle}.L \leqslant f_{j+1}^{s_i,\langle v_a,v_b\rangle}.\phi \end{aligned} \tag{3}$$

Additionally, the streams' end-to-end latency requirements should be satisfied:

$$\begin{aligned} &\forall s_i \in S: \\ &\quad \text{if } s_i.type = Det: \\ &\qquad f_{last}^{s_i,s_i.path[last]}.\phi - f_1^{s_i,s_i.path[0]}.\phi \leqslant s_i.e2e \\ &\quad \text{else}: \\ &\qquad f_{last}^{s_i,s_i.path[last]}.\phi - s_i.ot \leqslant s_i.e2e \end{aligned} \tag{4}$$

Looping over all the probabilistic streams of an ECT stream, (4) ensures that no matter when the ECT stream occurs, the TSN network can deliver its frames before the deadline.

2) Frame Overlap Constraints: A link can only transmit one frame at a time, so when scheduling TCT solely, the scheduled time-slots for two frames cannot overlap. However, in our algorithm, there are two conditions when the time-slots of frames can overlap: (1) when two frames belong to different probabilistic streams derived from the same ECT stream, since only one of them will actually exist as we discussed in Sec. III-B. (2) when one is a probabilistic stream, and the other is a TCT stream that shares time-slots, since the TCT stream's reserved time-slots have already been expanded in Alg. 1. These can be formalized as follows:

$$\begin{aligned} &\forall \langle v_a, v_b \rangle \in E: \\ &\forall F_{s_i,\langle v_a,v_b\rangle}, F_{s_j,\langle v_a,v_b\rangle}, i \neq j: \\ &\text{if } s_i \text{ and } s_j \text{ can not overlap}: \\ &\quad \forall f_k^{s_i,\langle v_a,v_b\rangle} \in F_{s_i,\langle v_a,v_b\rangle}, f_l^{s_j,\langle v_a,v_b\rangle} \in F_{s_j,\langle v_a,v_b\rangle}: \\ &\quad T^{hyper} \leftarrow lcm\left(s_i.T, s_j.T\right) \\ &\quad \forall x \in \{0, 1, ..., T^{hyper}/s_i.T - 1\}: \\ &\quad \forall y \in \{0, 1, ..., T^{hyper}/s_j.T - 1\}: \\ &\quad (f_k^{s_i,\langle v_a,v_b\rangle}.\phi + x \times f_k^{s_i,\langle v_a,v_b\rangle}.T \geqslant \\ &\quad f_l^{s_j,\langle v_a,v_b\rangle}.\phi + y \times f_l^{s_j,\langle v_a,v_b\rangle}.T + f_l^{s_j,\langle v_a,v_b\rangle}.L) \vee \\ &\quad (f_l^{s_j,\langle v_a,v_b\rangle}.\phi + y \times f_l^{s_j,\langle v_a,v_b\rangle}.T \geqslant \\ &\quad f_k^{s_i,\langle v_a,v_b\rangle}.\phi + x \times f_k^{s_i,\langle v_a,v_b\rangle}.T + f_k^{s_i,\langle v_a,v_b\rangle}.L) \end{aligned} \tag{5}$$

where $lcm(s_i.T, s_j.T)$ calculates the least common multiple of $s_i.T$ and $s_j.T$, *i.e.*, the hyperperiod of the two streams.

3) Priority Constraints: The priority determines in which queue the frames wait to be transmitted. This enables the spatial isolation of streams in switches. A TSN network can have at most eight priorities. We reserve one of them for ECT (EP), then divide the remaining priorities into two groups. One group, from SH_PL to SH_PH, is for TCT that shares time-slots. The other group, from NSH_PL to NSH_PH, is for TCT that does not share time-slots.

$$\begin{aligned} &\forall s_i \in S: \\ &(s_i.type = Prob \wedge s_i.p = EP) \vee \\ &(s_i.type = Det \wedge s_i.share = False \\ &\wedge s_i.p \geqslant NSH\_PL \wedge s_i.p \leqslant NSH\_PH) \vee \\ &(s_i.type = Det \wedge s_i.share = True \\ &\wedge s_i.p \geqslant SH\_PL \wedge s_i.p \leqslant SH\_PH) \end{aligned} \tag{6}$$

ECT is not always prioritised w.r.t. TCT. TCT streams that have tight latency bounds can be assigned a priority in the second group, *i.e.*, from NSH_PL to NSH_PH, so they won't be affected by ECT.

The attribute $s_i.share$ can be specified manually according to the importance of the streams. It can also be a variable, thus let the algorithm determine if s_i can share time-slots based on the overall scheduling requirements.

4) Adjacent Link Constraints: In Sec. III-D, we have discussed that the time-slots on the downstream link should be scheduled after the latest slot on the upstream link that may transmit the same frame. This can be formalized as:

$$\begin{aligned} &\forall s_i \in S, \forall \langle v_a, v_b \rangle, \langle v_b, v_c \rangle \in s_i.path: \\ &o \leftarrow Max(\left|F_{s_i,\langle v_a,v_b\rangle}\right| - \left|F_{s_i,\langle v_b,v_c\rangle}\right|, 0) \\ &\forall f_j^{s_i,\langle v_b,v_c\rangle} \in F_{s_i,\langle v_b,v_c\rangle}: \\ &f_j^{s_i,\langle v_b,v_c\rangle}.phi \times \langle v_b, v_c \rangle.tu - \langle v_a, v_b \rangle.d \geqslant \\ &(f_{j+o}^{s_i,\langle v_a,v_b\rangle}.phi + f_{j+o}^{s_i,\langle v_a,v_b\rangle}.L) \times \langle v_a, v_b \rangle.tu \end{aligned} \tag{7}$$

(7) also guarantees that (4) models the worst-case latency of the TCT streams.

The constraints in this section formulate the joint scheduling of ECT and TCT as an SMT problem. It can then be solved by SMT solver like z3 [19].

V. Testbed Implementation

To support the evaluation of E-TSN and further TSN studies, we develop TSN platform Ziggo from scratch, which includes both TSN switches and evaluation toolkits. Ziggo complies with IEEE TSN standards and can help fill the gap between algorithm design and experimental validation in TSN study. Both switches and toolkits are implemented on Xilinx ZYNQ-7000 SoC [20] (board model AX7021 [21]). ZYNQ-7000 SoC integrates the software programmability of an ARM processor (a.k.a PS, processor system) with the hardware programmability of an FPGA (a.k.a. PL, programmable logic).

The overview of TSN switches and evaluation tools' design is illustrated in Fig. 9. The TSN switches and evaluation toolkits support 802.1Qbv [15], 802.1AS [13], as well as 802.1Qcc [16]. The timestamps are obtained in hardware to

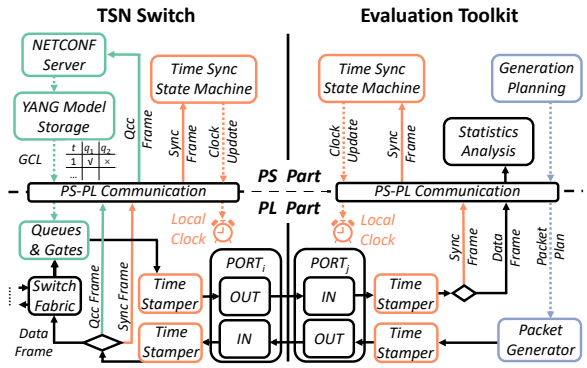


Fig. 9: Overview of TSN switch and evaluation tools' design.

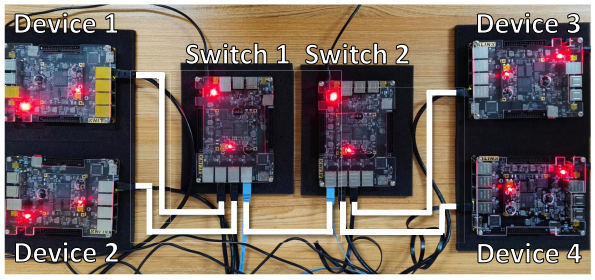


Fig. 10: TSN Testbed with two switches and four devices

achieve 10ns accuracy. The frames of TSN configuration (Qcc) and time synchronization (802.1AS) are forwarded from PL to PS through Direct Memory Access (DMA). Other frames are transmitted by hardware logic without CPU processing. Modules like gate control list, switch fabric, and transmission queue selection are also implemented in pure hardware. The Precision Time Protocol (802.1AS), NETCONF protocol, and YANG model (Qcc) are implemented in software, and they access hardware components via DMA or memory-mapped registers.

Fig. 10 shows our testbed with two switches and four devices (evaluation toolkit). We will compare the performance of E-TSN and other methods in this testbed in Sec. VI-B.

VI. Evaluation

We compare the performance of E-TSN and other methods in a TSN testbed and a TSN simulation tool. The experiments are done with different network load and traffic settings. We measure the latency and jitter of ECT, as well as its impact on TCT. Our experiments reveal the following key findings.

- Both in testbeds and simulations, E-TSN achieves at least one order of magnitude lower latency and jitter for ECT streams compared to other methods. E-TSN can provide 515us worst-case latency in a 3-hops TSN testbed.
- The performance of E-TSN is robust in various settings, including different network loads and different payload lengths.
- E-TSN allows ECT streams to share the time-slots of TCT streams. E-TSN is aware of this impact and can always guarantee that the network requirements of TCT streams are satisfied even in the worst case.
- When there are multiple ECT streams, E-TSN can provide an order of magnitude lower latency and jitter to all the streams.

In the following, we first introduce the evaluation methodology in Sec. VI-A. Then we report the results in the testbed in Sec. VI-B. At last, we analyze the performance of E-TSN theoretically in Sec. VI-C.

A. Methodology

1) Evaluation Setting: We compare E-TSN with other methods in both a TSN testbed and a TSN simulation platform. The simulation is to validate the performance of E-TSN in a larger scale network and its compatibility with TSN standards. The testbed has been introduced in Sec. V. The simulator we use is NeSTiNg [22], which is based on OMNeT++ [23]. It is also used by IEEE 802.1 Working Group for the development of new standards [24]. The network topology, traffic specifications, and evaluation results of testbeds and simulations will be reported in Sec. VI-B and Sec. VI-C.

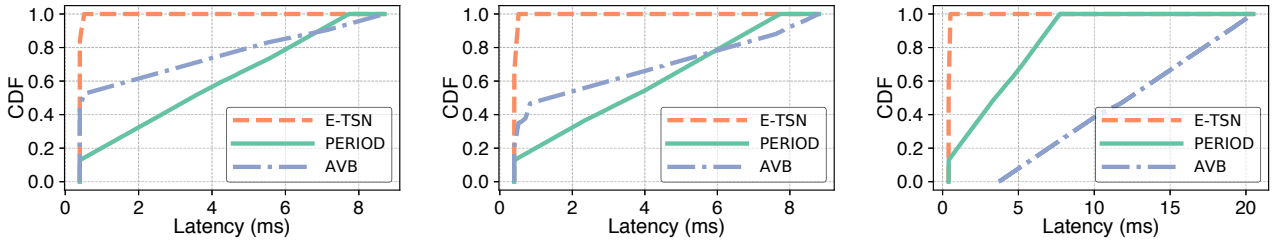
2) Comparing Methods: Since there is no previous work on enabling ECT in TSN, we compare E-TSN with two methods that are based on state-of-the-art traditional TSN scheduling algorithms:

- **PERIOD:** An intuitive idea is to treat ECT as TCT, then solve the scheduling problem using TCT scheduling algorithm such as [8]. We allow PERIOD to schedule ECT with a period smaller than its minimum interevent time to achieve lower latency. We make PERIOD use as many time-slots as E-TSN if not otherwise stated.
- **AVB:** Another solution is to transmit ECT as Audio-Video-Bridge traffic defined in 802.1Qav [14]. This means that ECT can only transmit in unallocated time-slots but with a higher priority than background traffic.

3) Evaluation Metrics: We measure the latency of both ECT and TCT. The latency is defined as the time between the receiving of the last frame and the sending of the first frame. We compare the average latency, worst-case latency, and standard deviation of latency (jitter) of different methods.

B. Testbed Results

The network topology of the testbed is shown in Fig. 10, and the link speed is 100Mbps. The network traffic is randomly generated in accordance with IEC/IEEE 60802 [25], which is the TSN profile for industrial automation. We generate ten periodic TCT streams. The source and destination are randomly chosen from the four devices. The period is randomly chosen from the set {4ms, 8ms, 16ms}. The payload length of the streams is adjusted to form different network load status. We send an ECT stream from Device 2 to Device 4 in Fig. 10, and assume that it can share the time-slots of all the TCT streams. The length of the message is one Ethernet MTU (Maximum Transmission Unit). The minimum interevent time is 16ms.



(a) 25% Network Load (b) 50% Network Load (c) 75% Network Load

Fig. 11: CDFs of ECT streams' latency using different scheduling algorithms and under different network loads. The network load means how much bandwidth is consumed by TCT in the network.

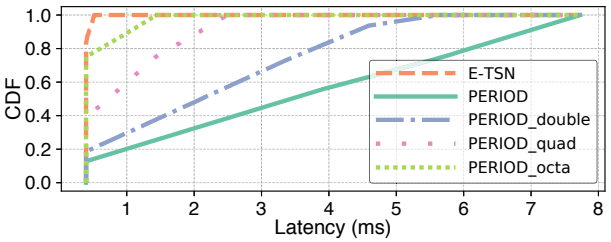


Fig. 12: CDFs of ECT streams' latency. PERIOD, PERIOD_double, PERIOD_quad, PERIOD_octa reserve 1, 2, 4, and 8 times as many time-slots as E-TSN.

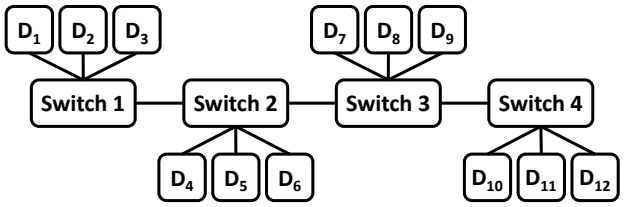


Fig. 13: Simulation network topology with four switches and twelve devices

The occurrence time of this stream is stochastic, in line with uniform distribution.

In the first experiment, we measure the latency of ECT under different methods' schedules, and the results are shown in Fig. 11. The network load means how much bandwidth is consumed by TCT in the network. The CDFs show that E-TSN can provide much lower latency and jitter to ECT. With 75% network load, the average latency of E-TSN is 423μs, which is 88% lower than that of PERIOD and 97% lower than that of AVB. The worst-case latency of E-TSN is 515us, which is more than an order of magnitude lower than that of PERIOD and AVB. We measure jitter using the standard deviation of latency. E-TSN's jitter is 39μs, which is two orders of magnitude lower than that of the other two methods. E-TSN allows ECT to transmit immediately when they occur. PERIOD allocates dedicated time-slots for ECT. Since the message can arrive at any time, PERIOD may need the message to wait for an entire period to transmit in the worst case.

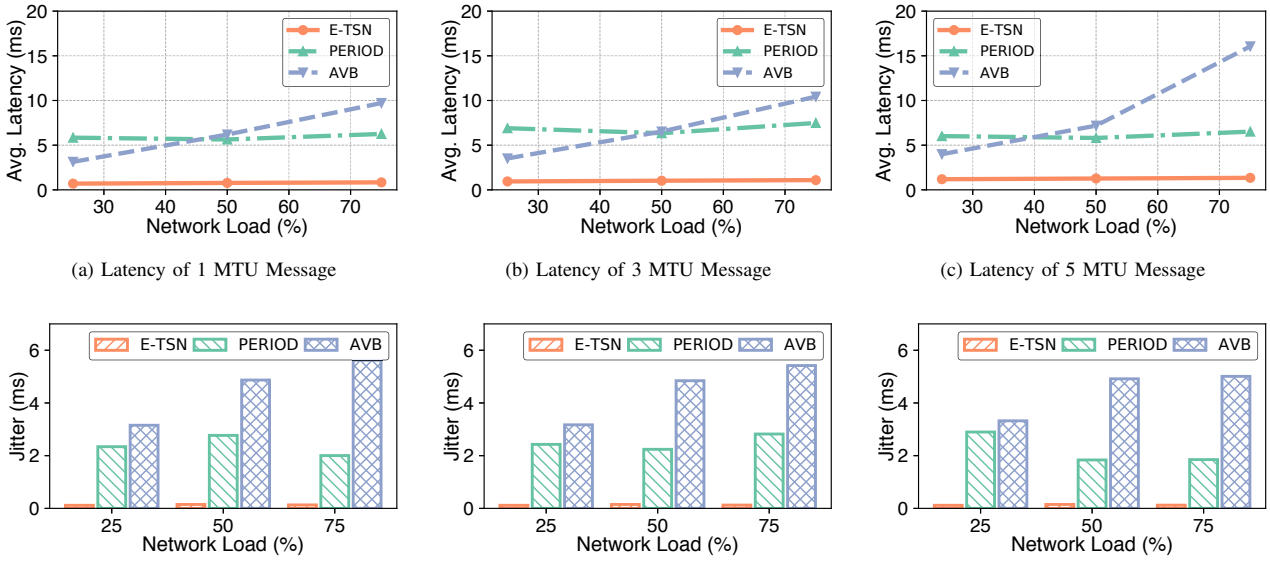
Another observation is that under different network loads, the performance of E-TSN and PERIOD is stable. This is because that E-TSN allows ECT to share the time-slots of TCT, and the exclusive time-slots in PERIOD are not influenced by the other traffic. However, the performance of AVB decreases rapidly with increasing network loads. From 25% network load to 75%, the average latency of AVB rises fivefold. This is because that AVB only allows ECT to transmit in unallocated time-slots. And the unallocated time slots may not be aligned between switches. So when the bandwidth allocated to TCT increases, AVB's performance decreases rapidly.

In the second experiment, we compare the resource cost of E-TSN compared to PERIOD. We allow PERIOD to reserve two to eight times as many time-slots as E-TSN for ECT. The results are shown in Fig. 12. Even with eight-times time-slots, the worst-case latency of PERIOD is still three times that of E-TSN. In this case, PERIOD consumes over 90% network bandwidth solely to transmit the ECT stream, which is impractical.

C. Simulation Results

The network topology for simulation is shown in Fig. 13, and the link speed is also 100Mbps. It consists of 4 switches and 12 devices. Forty TCT streams are generated based on IEC/IEEE 60802. The period is chosen from {5ms, 10ms, 20ms}, and the payload is adjusted to form different network loads. An ECT stream is sent from Device 1 to Device 12. Its minimum interevent time is 10ms. Its occurrence time is in line with uniform distribution. We assume that it can share the time-slots of all the TCT streams, except in Sec. VI-C2.

1) Latency and jitter of ECT streams: Like in Sec. VI-B, we first measure the latency of ECT with different network loads. We also change the length of the ECT stream from one MTU to five MTU to see its impact on the latency and jitter. As shown in Fig. 14(a)(b)(c), the latency of E-TSN is consistently the lowest under different settings. It is not affected by the network load or the message length. On average, the latency of E-TSN is 83.8% lower than PERIOD, and 83.1% lower



(a) Latency of 1 MTU Message (b) Latency of 3 MTU Message (c) Latency of 5 MTU Message

(d) Jitter of 1 MTU Message (e) Jitter of 3 MTU Message (f) Jitter of 5 MTU Message

Fig. 14: (a)(b)(c) show the latency of ECT with different network loads and message length. (d)(e)(f) show the respective jitter. Jitter is calculated as the standard deviation of the latency.

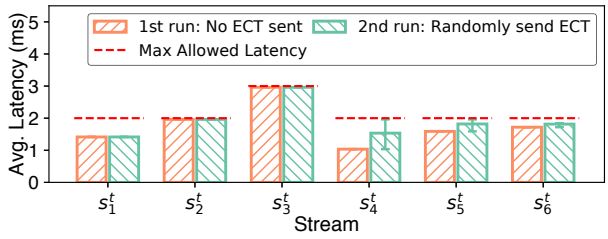


Fig. 15: s_4^t, s_5^t, and s_6^t share their time-slots with ECT, while s_1^t, s_2^t, and s_3^t do not. In the first run, we do not transmit ECT, and in the second run, we generate ECT randomly. The error bar is from the minimum latency to the maximum latency.

than AVB; the worst-case latency of E-TSN is 88.5% lower than PERIOD, and 91.7% lower than AVB; and the jitter of E-TSN is 94.3% lower than PERIOD, and 97.0% lower than AVB;

When the network load increases from 25% to 75%, the performance of E-TSN and PERIOD is not affected, while the performance of AVB degrades rapidly, similar to what we have observed in the testbed (Sec. VI-B). On the other hand, when the length of the ECT message increases from 1 MTU to 5 MTU, the latency of E-TSN and PERIOD only increases slightly, while the latency of AVB increases greatly. This is because E-TSN and PERIOD explicitly reserve extra shared or dedicated time-slots that are enough to transmit the entire ECT message.

2) Impacts on TCT streams: E-TSN allows ECT to share the time-slots of TCT. This may increase the latency and jitter of TCT streams. To study the impact of ECT on TCT in E-TSN, we measure the latency of TCT streams with or without ECT. We assume that ten out of forty TCT streams are more important than ECT, and they do not share time-slots. The results are shown in Fig. 15. The latency of three shared TCT streams and three non-shared streams are presented. The results for other streams are similar.

The results show that E-TSN can protect TCT from the encroachment of ECT and guarantee that the requirements of TCT are always satisfied even in the worst case. For streams that do not share time-slots, *i.e.*, s_1^t, s_2^t, and s_3^t, the presence of ECT makes no difference. For streams that share time-slots, *i.e.*, s_4^t, s_5^t, and s_6^t, although their latency and jitter increase when there is randomly generated ECT, the worst-case latency is always below the maximum allowed latency of the respective streams.

3) Multiple ECT streams: We also measure the performance of different methods when there are multiple ECT streams in the network. The network load is set to 50%. Besides the stream from Device 1 to Device 12 (s_1^e), we create three other ECT streams, *i.e.*, s_2^e, s_3^e, and s_4^e. The source and destination devices are chosen randomly from the 12 devices. All of the four streams send traffic at a random time in line with uniform distribution.

The latency and jitter of the four streams using different methods are shown in Fig. 16. E-TSN achieves the lowest latency and jitter for all the streams. On average, E-TSN can reduce the latency by 85.4% and the jitter by 97.0% compared to AVB, and reduce the latency by 78.7% and the jitter by

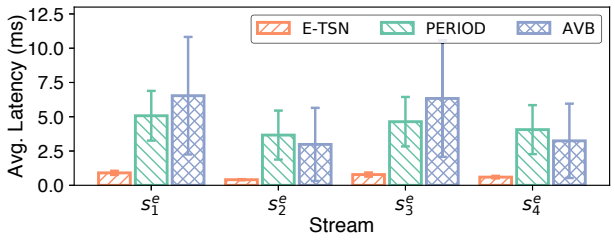


Fig. 16: Latency and jitter of four ECT streams: s_1^e, s_2^e, s_3^e, and s_4^e. The length of the error bar is double standard deviation of the latency.

93.7% compared to PERIOD.

VII. Related Work

A. Evolution of TSN

The IEEE TSN Task Group (TG) was evolved from the former Audio Video Bridging (AVB) TG. AVB concentrates on the QoS of audio/video traffic inside an Ethernet network [26]. In 2012, AVG TG was renamed to TSN TG, and the focus of the group shifted from audio/video services to industrial real-time communications. Numerous techniques have been introduced to realize deterministic transmission on standard Ethernet, such as Precision Time Protocol [13], Time Aware Shaper [15], and Cyclic Queuing and Forwarding [27]. Other techniques like SDN [28], OPC UA [2], and seamless redundancy [29] are also explored to automate the configuration process and provide extra reliability. The above techniques and corresponding scheduling algorithms have enabled the deterministic transmission of TCT in TSN. However, they do not support ECT, which limits the scope of application of TSN in industry. In this work, we propose a new scheduling paradigm for TSN, and for the first time, discuss and solve the problem of enabling ECT in TSN.

Next, we discuss previous work of offline and online scheduling for TCT in TSN.

B. Offline TSN Scheduling

This category of previous work focuses on the offline scheduling prior to the operation of the network. So the scheduling algorithm does not need to consider the dynamic change of the network in operation. IEEE 802.1Qbv was published in 2015, and since then, many researchers have studied the scheduling problem of it. Craciunas *et al.* first use Satisfiability Modulo Theories (SMT) to model the 802.1Qbv scheduling problem [8], based on their prior studies on Time-Triggered Ethernet (TTE) [30]–[32]. They also discussed the pros and cons of flow isolation and frame isolation strategies. Later, they formalize the scheduling problem as a system of constraints expressed using *first-order theory of arrays* [9], which is then solved using SMT solvers. Besides SMT, another trend is to formulate the scheduling problem as Integer Linear Programs (ILP) [33]–[35]. Nayak *et al.* show that the well-known *No-wait Job-shop Scheduling Problem* (NW-JSP) can be adapted to calculate TSN schedules [36]. Pop *et al.* and Steiner *et al.* also take the scheduling of mixed-criticality traffic into consideration [11]. Besides sole traffic scheduling, [35] takes traffic routing into consideration and proposes an ILP-based joint routing and scheduling algorithm. However, all the above work focuses on the scheduling of TCT without considering the existence of ECT. In this work, we solve the scheduling problem of ECT without violating TCT's requirements.

C. Online TSN Scheduling

The network topology and data transmission requirements are not always static, since network links may fail and the upper applications may change. Different from offline scheduling, online scheduling focuses on the speed of the scheduling algorithm [37]–[39]. So the network can accommodate to the dynamic change of networks and streams by updating the schedule in real-time. Yan *et al.* propose a novel algorithm based on Tabu search [40] to schedule TCT flows in CQF-based TSN, which improves the mapped flow number by 10x [10]. Nayak *et al.* also design a Tabu-based heuristic algorithm, which split the scheduling into a time-tabling problem and a sequencing problem [36]. Besides the widely used Tabu heuristic, Pop *et al.* design an algorithm based on Greedy Randomized Adaptive Search Procedure (GRASP) [11], [41]. And Steiner proposes an incremental backtracking algorithm in [18] to accelerate the solving of SMT. Recently, with the huge success of artificial intelligence, some prior work studies the application of deep learning to TSN scheduling problem. Zhong *et al.* propose a Deep Reinforcement Learning based scheduling method for time-triggered traffic, which shows better performance than traditional heuristic-based methods [42]. Previous work designs heuristic- or DNN-based algorithms for the scheduling of TCT in TSN. Some of these techniques can be directly used in our algorithm like [18]. Designing online scheduling algorithm for ECT is an interesting direction of our future work.

VIII. Conclusion

With the rise of Industry 4.0 and the Industrial Internet, TSN is gaining more and more attention. In this paper, we propose a new paradigm for TSN scheduling named E-TSN. It enables the reliable and timely transmission of ECT in TSN for the first time. We propose three novel techniques to model ECT's different possibilities, guarantee low latency no matter when ECT occurs, and ensure the fulfillment of TCT's network requirements. We also develop and make public a TSN evaluation toolkit on FPGA, which supports major TSN standards and measures network latency at 10ns accuracy. The experiments show that E-TSN achieves at least an order of magnitude lower latency and jitter for ECT compared to state-of-the-art methods. The ability to support both TCT and ECT can promote the application of TSN in more scenarios of manufacturing and automation in the future.

ACKNOWLEDGMENT

This work is supported in part by the National Key Research Plan under grant No. 2021YFB2900100, the NSFC under grant No. 61832010.

REFERENCES

[1] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," *Business & information systems engineering*, vol. 6, no. 4, pp. 239–242, 2014.

[2] D. Bruckner, M.-P. Stănică, R. Blair, S. Schriegel, S. Kehrer, M. Seewald, and T. Sauter, "An introduction to opc ua tsn for industrial communication systems," *Proceedings of the IEEE*, vol. 107, no. 6, pp. 1121–1131, 2019.

[3] J. Eklahare. (2021, Jul.) Why tsn is the future in manufacturing. [Online]. Available: https://www.industr.com/en/why-tsn-is-the-future-in-manufacturing-2590686

[4] A. Albert *et al.*, "Comparison of event-triggered and time-triggered concepts with regard to distributed control systems," *Embedded world*, vol. 2004, pp. 235–252, 2004.

[5] M. S. Mahmoud and Y. Xia, "Chapter 2 - Networked Control Systems' Fundamentals," in *Networked Control Systems*, M. S. Mahmoud and Y. Xia, Eds. Butterworth-Heinemann, 2019, pp. 37–89.

[6] X. Ge, Q.-L. Han, X.-M. Zhang, and D. Ding, "Dynamic event-triggered control and estimation: a survey," *International Journal of Automation and Computing*, pp. 1–30, 2021.

[7] T. Li, Q. Qiu, and C. Zhao, "A fully distributed event-triggered communication strategy for second-order multi-agent systems consensus," 2020.

[8] S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner, "Scheduling real-time communication in ieee 802.1qbv time sensitive networks," in *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, ser. RTNS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 183–192. [Online]. Available: https://doi.org/10.1145/2997465.2997470

[9] R. Serna Oliver, S. S. Craciunas, and W. Steiner, "Ieee 802.1qbv gate control list synthesis using array theory encoding," in *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2018, pp. 13–24.

[10] J. Yan, W. Quan, X. Jiang, and Z. Sun, "Injection time planning: Making cqf practical in time-sensitive networking," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 616–625.

[11] V. Gavriluţ and P. Pop, "Scheduling in time sensitive networks (tsn) for mixed-criticality industrial applications," in *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, 2018, pp. 1–4.

[12] C. Barrett and C. Tinelli, "Satisfiability modulo theories," in *Handbook of model checking*. Springer, 2018, pp. 305–343.

[13] *Timing and Synchronization for Time-Sensitive Applications*, IEEE Std. 802.1AS, 2020.

[14] *Forwarding and Queuing Enhancements for Time-Sensitive Streams*, IEEE Std. 802.1Qav, 2009.

[15] *Enhancements for Scheduled Traffic*, IEEE Std. 802.1Qbv, 2015.

[16] *Stream Reservation Protocols (SRP) Enhancements and Performance Improvements*, IEEE Std. 802.1Qcc, 2018.

[17] *Bridges and Bridged Networks*, IEEE Std. 802.1Q, 2018.

[18] W. Steiner, "An evaluation of smt-based schedule synthesis for time-triggered multi-hop networks," in *2010 31st IEEE Real-Time Systems Symposium*, 2010, pp. 375–384.

[19] Z3Prover. (2021, Jul.) Github repository of z3prover. [Online]. Available: https://github.com/Z3Prover/z3

[20] Xilinx. (2021, Jul.) Socs with hardware and software programmability. [Online]. Available: https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html

[21] Alinx. (2021, Jul.) Alinx xilinx zynq fpga development board. [Online]. Available: http://alinx.com/index.php/default/content/96.html

[22] J. Falk, D. Hellmanns, B. Carabelli, N. Nayak, F. Dürr, S. Kehrer, and K. Rothermel, "Nesting: Simulating ieee time-sensitive networking (tsn) in omnet++," in *2019 International Conference on Networked Systems (NetSys)*, 2019, pp. 1–8.

[23] O. Group. (2021, Jul.) Omnet++ is an extensible, modular, component-based c++ simulation library and framework, primarily for building network simulators. [Online]. Available: https://omnetpp.org/

[24] IEEE. (2021, Jul.) Ieee 802.1 protocol simulations. [Online]. Available: https://omnetpp.org/

[25] *TSN Profile for Industrial Automation*, IEC/IEEE Std. 60 802, 2018.

[26] Wiki. (2021, Jul.) Audio video bridging. [Online]. Available: https://en.wikipedia.org/wiki/Audio_Video_Bridging

[27] *Cyclic Queuing and Forwarding*, IEEE Std. 802.1Qch, 2017.

[28] S. B. H. Said, Q. H. Truong, and M. Boc, "Sdn-based configuration solution for ieee 802.1 time sensitive networking (tsn)," *ACM SIGBED Review*, vol. 16, no. 1, pp. 27–32, 2019.

[29] *Frame Replication and Elimination for Reliability*, IEEE Std. 802.1CB, 2017.

[30] F. Pozo, G. Rodriguez-Navas, H. Hansson, and W. Steiner, "Smt-based synthesis of ttethernet schedules: A performance study," in *10th IEEE International Symposium on Industrial Embedded Systems (SIES)*. IEEE, 2015, pp. 1–4.

[31] F. Pozo, W. Steiner, G. Rodriguez-Navas, and H. Hansson, "A decomposition approach for smt-based schedule synthesis for time-triggered networks," in *2015 IEEE 20th conference on emerging technologies & factory automation (ETFA)*. IEEE, 2015, pp. 1–8.

[32] S. S. Craciunas and R. S. Oliver, "Combined task-and network-level scheduling for distributed time-triggered systems," *Real-Time Systems*, vol. 52, no. 2, pp. 161–200, 2016.

[33] P. Pop, M. L. Raagaard, M. Gutierrez, and W. Steiner, "Enabling fog computing for industrial automation through time-sensitive networking (tsn)," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 55–61, 2018.

[34] J. Falk, F. Dürr, and K. Rothermel, "Exploring practical limitations of joint routing and scheduling for tsn with ilp," in *2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. IEEE, 2018, pp. 136–146.

[35] E. Schweissguth, P. Danielis, D. Timmermann, H. Parzyjegla, and G. Mühl, "Ilp-based joint routing and scheduling for time-triggered networks," in *Proceedings of the 25th International Conference on Real-Time Networks and Systems*, 2017, pp. 8–17.

[36] F. Dürr and N. G. Nayak, "No-wait packet scheduling for ieee time-sensitive networks (tsn)," in *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, ser. RTNS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 203–212. [Online]. Available: https://doi.org/10.1145/2997465.2997494

[37] A. A. Atallah, G. B. Hamad, and O. A. Mohamed, "Fault-resilient topology planning and traffic configuration for ieee 802.1 qbv tsn networks," in *2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS)*. IEEE, 2018, pp. 151–156.

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

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

[40] F. Glover and M. Laguna, "Tabu search," in *Handbook of combinatorial optimization*. Springer, 1998, pp. 2093–2229.

[41] M. G. Resende and C. C. Ribeiro, "Grasp: Greedy randomized adaptive search procedures," in *Search methodologies*. Springer, 2014, pp. 287–312.

[42] C. Zhong, H. Jia, H. Wan, and X. Zhao, "Drls: A deep reinforcement learning based scheduler for time-triggered ethernet," in *2021 International Conference on Computer Communications and Networks (ICCCN)*, 2021, pp. 1–11.