ADA: Asynchronous Deterministic Access for Time-Sensitive Networking

Cheng Long, Wenlin Zhu, Zixiao Wang, Yiming Zheng, Zonghui Li, and Kang G. Shin, *Life Fellow, IEEE*

***Abstract*—Time-Sensitive Networking (TSN) enables deterministic transmission by requiring end systems to support 802.1Qbv and 802.1AS standards, which makes practical deployment challenging, especially for asynchronous end systems (AES) such as various legacy devices already deployed in industry. We present a novel deterministic transmission scheme for asynchronous time-triggered flows from AES. First, we design an Asynchronous Deterministic Access (ADA) mechanism with two components: "Async to Sync" and "Latency Adaptation." These components allow an AES flow to wait for its scheduled period, ensuring a theoretical end-to-end (E2E) deterministic delay. Second, we propose an ADA scheduling model to optimize the scheduling periods for AES flows. However, latency adaptation may cause AES flows to conflict with other flows at the final hop switch. To address this problem, we propose a conflict resolution mechanism that determines safety distances between flows to reduce delay jitter. Finally, we implement ADA in our switches and evaluate its performance in terms of E2E delay and jitter. ADA is shown to achieve E2E delay and jitter reductions of 57.1–83.5% and 60.7–85.4%, respectively, over SOTA methods.**

***Index Terms*—Asynchronous end system (AES), time-triggered, time-sensitive networking (TSN).**

I. Introduction

OVER the past decade, digital transformation has driven the development of highly intelligent, networked, and flexible industrial automation systems. Evolving from the traditional Ethernet, Time-Sensitive Networking (TSN) is a new and promising industrial communication protocol standardized by the IEEE 802.1 TSN Group to enable deterministic and real-time Ethernet-based communication [1], [2], [3]. TSN provides deterministic latency, low jitter, high efficiency, and high reliability, supporting E2E industrial communication from manufacturing lines to industrial clouds.

To achieve real-time and deterministic communication, TSN specifies the Time-Aware Shaper (TAS) gate mechanism in IEEE 802.1Qbv [4], enabling time-triggered (TT) communication. TAS uses the Gate Control List (GCL) in each switch to precisely control the periodic transmission instants of flows. Flows transmitted through IEEE 802.1Qbv are referred to as *TT flows*.

The precise timing of gate control relies on network-wide time synchronization, requiring both switches and end systems to be synchronized [5]. However, in many industrial settings, switches are provided by network vendors, while end systems are provided by different manufacturers. As a result, a large number of unsynchronized end systems have already been deployed on factory floors. It would be too costly and time-consuming to replace or upgrade all of these end systems to TSN-compatible ones. Therefore, it is highly unlikely to upgrade network and end systems simultaneously, leaving end systems, especially legacy devices, TSN-unsynchronized. Such unsynchronized end systems are referred to as *asynchronous end systems* (AES). TT flows transmitted by AES devices are called *AES flows*.

When an AES device transmits AES flows, their frames/packets may not be transmitted exactly at their scheduled time by the first switch (the one directly connected to the AES device) in the network. As a result, they are not forwarded to the next switch/device until the next scheduled transmission time in the following period. The resulting additional delays and jitters, up to one period per flow, degrade the deterministic transmission capability of TSN. How to guarantee the deterministic E2E transmission for AES flows remains an open problem.

Existing methods for addressing asynchronous transmission issues are typically classified into three types. The first type transmits AES flows as synchronized flows using time-triggered scheduling algorithms, without addressing the AES access issues [6]. While this type makes AES flows compatible with synchronized schedules, it causes additional delay and jitter up to their own scheduled period. The second type converts AES flows to bursty flows and shapes them using asynchronous traffic shaping mechanisms like IEEE 802.1Qav Credit Based Shaper (CBS) [7] and IEEE 802.1Qcr Asynchronous Traffic Shaper (ATS) [8]. Although this type can establish an upper delay bound based on current parameter

Received 24 January 2025; revised 9 August 2025 and 20 November 2025; accepted 16 February 2026; approved by IEEE TRANSACTIONS ON NETWORKING Editor M. Levorato. Date of publication 24 February 2026; date of current version 2 March 2026. This work was supported in part by the National Natural Science Foundation of China under Grant 62531003, Grant 62272034; and in part by the Science and Technology Project of State Grid Corporation of China under Grant 52120025003R-295-LZ. *(Corresponding author: Zonghui Li.)*

Cheng Long, Zixiao Wang, and Zonghui Li are with the School of Computer Science and Technology, Beijing Jiaotong University, Beijing 100044, China (e-mail: long.c@bjtu.edu.cn; w_zixiao@bjtu.edu.cn; zonghui.lee@gmail.com).

Wenlin Zhu is with China United Network Communications Company Ltd., Linyi Branch, Linyi 276002, China (e-mail: zhuwenlin@bjtu.edu.cn).

Yiming Zheng is with the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China (e-mail: yiming_z@bjtu.edu.cn).

Kang G. Shin is with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: kgshin@umich.edu).

Digital Object Identifier 10.1109/TON.2026.3667819

settings, it lacks flexibility for custom delay requirements and usually yields very pessimistic delay upper bounds. Consequently, it does not effectively control latency jitter. The third type combines synchronous and asynchronous methods. They expand the synchronous transmission times into time windows to reduce the probability of missing them and perform a worst-case analysis for AES flows within these windows [9], [10], [11]. As a result, AES flows will still miss the time window, causing additional delay jitter due to waiting for the next sending window. In summary, existing approaches have not yet solved the deterministic transmission problem for AES flows.

We propose a new approach called the *Asynchronous Deterministic Access* (ADA), effectively addressing the challenge of asynchronous end system access and achieving deterministic transmission for AES flows. Unlike existing methods that either synchronize asynchronous flows or shape them using asynchronous mechanisms, ADA introduces a latency-compensation principle at the edges of the TSN domain. By coordinating a wait time at the ingress switch and the corresponding hold time at the egress switch, ADA ensures a constant and predictable E2E delay for each AES flow. This paper makes the following main contributions:

- **Design of a novel mechanism, ADA, to ensure the deterministic transmission of AES flows,** including: (1) "Async to Sync" and "Latency Adaptation" ensure the determinism of AES flow transmission and consolidate potential conflicts at the final hop; (2) "Scheduling Period" ensures that AES flows encounter their transmission instants more frequently, enhancing real-time performance; (3) "Conflict Resolution" calculates safety distances between flows to reduce jitter.
- **Development of an ADA scheduling model** that determines the scheduling periods, transmission instants for AES flows at each hop within the synchronized domain, and safety distances to ensure conflict-free transmission.
- **Evaluation of ADA's performance in comparison with the state-of-the-art (SOTA) methods and validation of its effectiveness on testbeds under real-world conditions.** Our evaluation results show that ADA is both feasible and effective, achieving deterministic packet delivery, 57.1–83.5% lower latencies, and 60.7–85.4% lower jitters than SOTA methods.

The rest of the paper is organized as follows. Related work is surveyed in Sec. II, and the problem statement is given in Sec. III. ADA is detailed in Sec. IV, followed by the ADA scheduling model in Sec. V. Our evaluation results and discussions are provided in Sec. VI and the conclusion in Sec. VII.

II. Related Work

TSN establishes the E2E deterministic transmission by forcing end systems to inject TT flows at the network-wide precise time instants [12]. As a result, AES flows may miss these precise transmission instants and lead to the extra delay jitter up to their own scheduled periods [13]. The prior work targeting such a problem can be classified into three groups as follows.

AES flows are treated as synchronous flows. There has been a large body of work on synchronous flows, including software approaches like TT schedulers [5], [14], [15], [16], [17], [18], [19], [20], [21], [22] to determine and assign precise transmission instants for time-critical data flows, hardware approaches like standardized logic architecture (i.e., IEEE 802.1Qbv [4], IEEE 802.1 Qch [23]) for TT transmission, TT switch, and end system architectures for resource-efficient [12], [13], [24], [25] and customizable [26], [27] implementations. The advantages of AES flows, as synchronous flows, are that all the methods developed for synchronous flows are applicable. But AES flows will result in the extra delay jitter up to their own scheduled periods. Reference [6] decreases the extra delay jitter by arranging more sending instants for each AES flow to lessen the scheduled periods. Such a method does not change the fact that AES flows cause the extra delay jitter up to their own scheduled periods.

AES flows are treated as asynchronous flows. Except for TT transmission standards (i.e., IEEE 802.1Qbv [4] and IEEE 802.1 Qch [23]), TSN also standardizes other traffic shaping and priority scheduling policies for asynchronous flows. IEEE 802.1Qav [7] defines a credit-based shaper (CBS) for real-time transmission of flows of different priority classes, typically Class A and Class B, by decreasing and/or increasing credit rates. References [28] and [29] used network calculus to analyze the worst E2E delay of different priority flows under varying credit rates. IEEE 802.1Qcr [8] defines an asynchronous traffic shaping (ATS) policy. That is, per-flow queue employs a token bucket shaper to eliminate the cascaded burstiness effects. The frames coming out of per-flow queues are then placed into different priority queues, called *per-class queues*, according to application-specific delay requirements, i.e., the more urgent the delay, the higher the priority. Finally, a fixed-priority scheduler ensures that higher-priority flows will be transmitted before lower-priority flows. References [24] and [25] designs a resource-efficient priority scheduler for embedded devices using a *flattened priority* framework to transform a non-priority scheduler, such as the classic iSLIP scheduler, into a priority scheduler. References [30] and [31] analyzed the worst-case E2E delay of frames under different numbers of per-flow queues and per-class queues. These traffic shaping and priority scheduling methods can only provide an upper bound of E2E delay and cannot customize the E2E delay for each flow, which may lead to unsatisfiable deadlines of flows. Reference [32] quantitatively analyzed the delay jitters of these traffic shapers using network calculus and demonstrated millisecond-level jitters, which weakens/degrades the determinacy of transmission.

AES flows are transmitted by combining synchronous and asynchronous methods. References [9], [10], and [11] proposed a flexible window-based GCL method (FWND) to decrease the possibility of asynchronous flows missing transmission chance by enlarging the precise transmission instants into sending windows of different sizes. Asynchronous flows falling within those windows compete for transmission according to priority and must wait for the next window if they miss the current one. As a result, the extra delay jitter includes in-window competition delay for transmission and

the scheduled period of a sending window, thus still causing millisecond-level delay jitters in the worst case. To achieve nearly zero jitter for AES flows, [33] employed buffers at the network egresses to end devices, to hold them until the worst-case network latency. Unfortunately, the worst-case network latency analysis [32] is usually very pessimistic. Such a long wait latency will increase the E2E delay significantly, missing the deadlines of AES flows. Instead of the worst-case latency analysis, [34] enhances the cyclic queueing and forwarding (CQF) of two queues in IEEE 802.1Qch into multiple cyclic queues, called *EM-CQF*. When AES flows miss the current sending queue, they are stored in other queues and wait for the next cyclic forwarding. For n cyclic queues, the delay jitter is $n\times$(the cyclic period). Since the transmission of mixed AES and synchronous flows to avoid interfering with each other, the smallest number of queues in EM-CQF is 3, and thus the delay jitter is $3\times$(the cyclic period).

Unlike all of the above methods, ADA employs two components — namely "Async to Sync" and "Latency Adaptation" — to establish the zero extra delay jitter for AES flows in theory by waiting for their own scheduled periods. We formalize the ADA model to get the optimized scheduling periods under their deadline constraints. So, ADA satisfies the deadlines of AES flows and avoids the pessimistic worst-case latency analysis of network calculus [32]. Due to "Latency Adaptation", AES flows may conflict with other flows. We propose a conflict resolution by calculating the safety distance between flows to decrease jitter. Note that the conflicts only happen in "Latency Adaptation". That is, ADA limits the conflicts to be possible only in the last switch directly linking to the receiving end system, unlike the network-wide conflicts of FWND. Our experimental results on typical network topologies demonstrate ADA to achieve 78.1%, 57.1% delay reductions and 85.6%, 58.2% jitter reductions over FWND and EM-CQF, respectively.

III. Problem Statement

A. System Model

The network topology is modeled as a directed graph $G = \{V, E\}$, where V represents the set of devices, including end systems (ES) and switches (SW). E represents the set of directed edges connecting pairs of devices. For each physical link between two devices v_a and v_b, there are two corresponding directed data links in opposite directions, $[v_a, v_b]$ and $[v_b, v_a] \in E$.

Let F denote the set of all TT flows transmitted in the network. For a network with k TT flows, we have $F = \{f_1, f_2, \ldots, f_k\}$. f_i includes the period, frame length, maximum transmission delay, transmission route, and configurable offsets at each hop. Formally, f_i can be represented as:

$$f_i = \{period, length, deadline, route, offsets\}. \quad (1)$$

The transmission route of f_i, denoted as $f_i.route$, consists of a sequence of directed edges and can be expressed as:

$$f_i.route = [[v_0, v_1], [v_1, v_2], \ldots, [v_{n-1}, v_n]]. \quad (2)$$

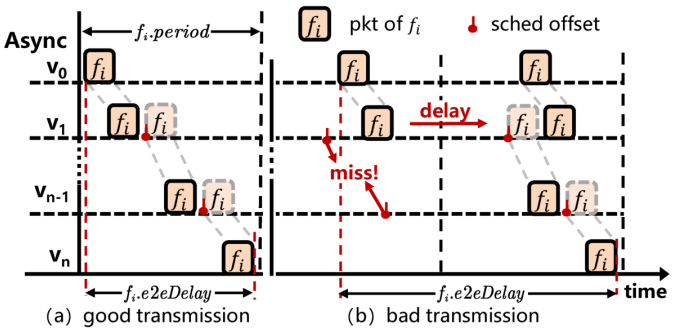


Fig. 1. Two scenarios for AES flow transmission in an AES device access.

The configurable offsets for f_i, denoted as $f_i.offsets$, is the set of transmission instants at each hop along the route. This can be expressed as:

$$f_i.offsets = \{f_i^{[v_i,v_j]}.offset | \forall [v_i, v_j] \in f_i.route\}. \quad (3)$$

Within the TSN domain, deterministic traffic (i.e., TT/AES flows) is transmitted in a time-triggered manner using the IEEE 802.1Qbv mechanism. Each egress port of a TSN switch maintains multiple priority queues corresponding to different traffic classes. Incoming frames are classified into these queues based on their Priority Code Point (PCP), using stream identification mechanisms defined in IEEE 802.1CB [35]. Additionally, IEEE 802.1Qci [36] provides per-stream filtering and policing at the ingress, enabling admission control and traffic shaping. In this context, TT/AES flows that require deterministic transmission are explicitly mapped to the TT queue——the highest-priority queue——and their transmission instants are enforced by pre-configured GCLs in the switches. Other types of traffic, such as AVB or Best-Effort (BE) flows, are classified into lower-priority queues.

B. Asynchronous End System Access

We explore the transmission of TT flows when AES devices access a synchronized domain. Switches within the synchronized domain are clock-synchronized, but they are not synchronized with the AES devices.

A TT flow f_i sent from an AES device is classified as an AES flow. Assume there is an AES flow f_i with route $f_i.route = [[v_0, v_1], [v_1, v_2], \ldots, [v_{n-1}, v_n]]$, where v_0 and v_n are AES, while $\{v_1, v_2, \ldots, v_{n-1}\}$ are clock-synchronized switches. The offsets for f_i are determined by the scheduler without considering time synchronization [5]. The transmission of f_i can result in two scenarios as shown in Fig. 1. Fig. 1a illustrates the case where f_i is transmitted normally: f_i arrives at switch v_1 from v_0 before its scheduled transmission instant at the next hop $[v_1, v_2]$, $f_i^{[v_1,v_2]}.offset$, and is transmitted normally at subsequent hops under synchronization. In this case, the E2E transmission delay of f_i achieves the optimal E2E delay, expressed as:

$$f_i.best = f_i^{[v_{n-1},v_n]}.offset - f_i^{[v_0,v_1]}.offset + f_i^{[v_{n-1},v_n]}.ld \quad (4)$$

where $f_i^{[v_{n-1},v_n]}.ld$ is the link delay of f_i on hop $[v_{n-1}, v_n]$.

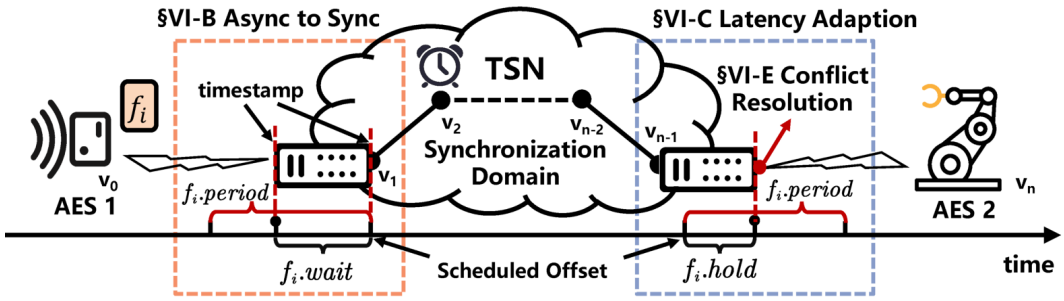


Fig. 2. Overview of ADA mechanism, including two component, "Async to Sync" and "Latency Adaptation". The extra delay sum of an AES flow in the two components is, in theory, equal to its scheduled period. That is, $f_i.wait + f_i.hold = f_i.period$. The "Conflict Resolution" ensures conflict-free transmission even with latency adaptation.

Fig. 1b illustrates the case of bad transmission for f_i. Due to the clock asynchrony between v_0 and v_1, f_i arrives at v_1 from v_0 later than $f_i^{[v_1,v_2]}.offset$. As a result, f_i must wait until the next period's transmission time at the hop $[v_1, v_2]$. In the worst case, f_i waits for an entire period at v_1, resulting in the worst-case transmission delay, expressed as:

$$f_i.worst = f_i^{[v_{n-1},v_n]}.offset - f_i^{[v_0,v_1]}.offset + f_i^{[v_{n-1},v_n]}.ld + f_i.period. \quad (5)$$

This may lead to a failure to meet the real-time requirement.

According to (4) and (5), we can derive the delay jitter for f_i which is equal to the difference between the maximum and minimum E2E delays:

$$f_i.jitter = f_i.worst - f_i.best = f_i.period. \quad (6)$$

Without proper guarantees, the access of the AES device may neither meet the real-time requirement nor achieve the deterministic transmission for TT flows. Thus, we need a mechanism that both meets the real-time requirements and achieves deterministic transmission for AES flows.

IV. Asynchronous Deterministic Access Mechanism

A. Workflow of Asynchronous Deterministic Access

Fig. 2 shows an overview of ADA. An AES device accesses the synchronized domain, periodically sending AES flows. Let f_i be an AES flow with $f_i.route$ defined as $[[v_0, v_1], [v_1, v_2], \ldots, [v_{n-1}, v_n]]$. ADA works as follows.

1) **Async to Sync.** The AES flow f_i from device v_0 enters the synchronized domain at the first switch v_1. The async-to-sync mechanism adjusts its transmission instant by waiting for the latest next transmission instant, making its transmission synchronous in the synchronized domain.
2) **Latency Adaptation.** At the last switch v_{n-1} of the synchronized domain, the latency adaptation mechanism ensures that f_i maintains a consistent E2E delay in each period, achieving deterministic transmission.
3) **Scheduling Period** ensures that AES flows encounter their transmission instants more frequently, thus reducing the E2E delay overhead and meeting real-time requirements.
4) **Conflict Resolution.** At the last switch v_{n-1}, the conflict resolution mechanism addresses the transmission conflicts caused by latency adaptation.

With the above mechanisms in place, deterministic transmission is achieved for AES access.

B. Async to Sync

Due to clock asynchrony, the arrival time of each AES flow at the first switch v_1 is unpredictable, which may cause the scheduled offset for the next hop to be missed. Let $f_i.wait$ represent the time that flow f_i waits after arriving at the first switch v_1 from the AES device v_0, before it gets forwarded to switch v_2 at the next scheduled transmission instant. The value of $f_i.wait$ can be obtained by calculating the difference between the timestamp when f_i is sent from v_1 and the timestamp when it arrives at v_1. The $f_i.wait$ value is then attached in the frame. After f_i waits $f_i.wait$ in switch v_1, f_i can continue to be transmitted within the synchronized domain using time-triggered transmission in sync. With the introduction of $f_i.wait$, the E2E delay of f_i can be expressed as:

$$f_i.e2eDelay = (f_i^{[v_{n-1},v_n]}.offset - f_i^{[v_1,v_2]}.offset) + f_i^{[v_0,v_1]}.ld + f_i^{[v_{n-1},v_n]}.ld + f_i.wait. \quad (7)$$

In the best case, $f_i.wait = 0$, meaning f_i arrives just before the next hop's scheduled transmission instant. In the worst case, f_i waits for an entire period, $f_i.wait = f_i.period$, having just missed the next hop's transmission instant. Since $f_i.wait \in [0, f_i.period]$, the jitter for f_i is:

$$f_i.jitter = f_i.period. \quad (8)$$

The async-to-sync mechanism introduces a wait time, allowing AES flows to align with the timing of the synchronized domain. This enables AES flows using time-triggered methods in the synchronous domain. However, this also increases the E2E delay by one period in the worst case, potentially compromising timeliness. Moreover, the jitter for AES flows is large, equal to one scheduled period, making it difficult to achieve deterministic transmission.

C. Latency Adaptation

To eliminate the impact of jitter, we introduce the latency adaptation mechanism. When the AES flow f_i reaches the last

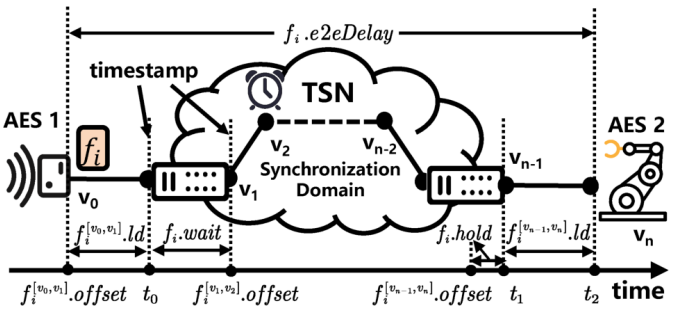


Fig. 3. The timing diagram for implementing the async-to-sync and latency-adaptation mechanisms.

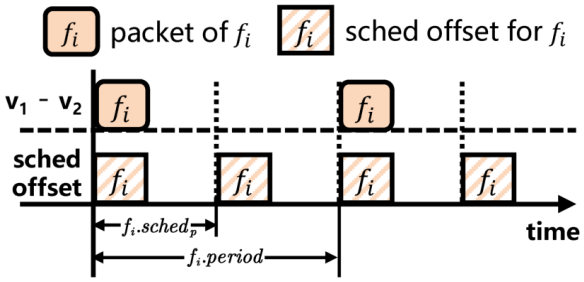


Fig. 4. Illustration of the relationship between the TT flow period and the scheduling period.

switch v_{n-1} of final hop $[v_{n-1}, v_n]$, it is not sent immediately. Instead, it is held for a certain amount of time, denoted as $f_i.hold$, at v_{n-1} before transmitting it. The value of $f_i.hold$ is determined by the difference between f_i's period $f_i.period$ and the wait time $f_i.wait$ at the first switch v_1. This can be expressed as:

$$f_i.hold = f_i.period - f_i.wait. \tag{9}$$

Fig. 3 shows the timing diagram for implementing the async-to-sync and latency adaptation mechanisms. In this case, the E2E delay of f_i becomes:

$$\begin{aligned} f_i.e2eDelay &= (f_i^{[v_{n-1},v_n]}.offset - f_i^{[v_1,v_2]}.offset) \\ &\quad + f_i^{[v_0,v_1]}.ld + f_i^{[v_{n-1},v_n]}.ld + f_i.wait + f_i.hold \\ &= (f_i^{[v_{n-1},v_n]}.offset - f_i^{[v_1,v_2]}.offset) \\ &\quad + f_i^{[v_0,v_1]}.ld + f_i^{[v_{n-1},v_n]}.ld + f_i.period. \end{aligned} \tag{10}$$

At this point, the E2E delay of f_i is constant. Therefore, the latency adaptation mechanism reduces the jitter of AES flows to 0, making it dependent solely on the accuracy of time synchronization, ensuring deterministic transmission. However, this comes at the expense of increasing the E2E delay by one scheduled period, which may fail to meet the real-time requirement of the AES flow.

D. Scheduling Period

TT flow f_i is transmitted from the source end system v_0 with period $f_i.period$. The transmission instant at each hop $[v_a, v_b]$ along the transmission route $f_i.route$ follows a periodic schedule with the same period as $f_i.period$. Let $f_i^{[v_a,v_b]}.offset_k$ denote the transmission instant of the k-th occurrence of f_i on the hop $[v_a, v_b]$, which can be expressed as:

$$f_i^{[v_a,v_b]}.offset_k = f_i^{[v_a,v_b]}.offset + (k-1) \cdot f_i.period. \tag{11}$$

As long as the reserved transmission slots of flow f_i on hop $[v_a, v_b]$ are configured to repeat at switch v_a with a frequency that divides $f_i.period$, each instance of f_i can always be served at its scheduled transmission instant without accumulating backlog at v_a.

We define the scheduling period of f_i as $f_i.sched_p$ and introduce a scheduling period mechanism. Fig. 4 illustrates the relationship between $f_i.period$ and $f_i.sched_p$, which satisfies the condition:

$$f_i.period \bmod f_i.sched_p = 0. \tag{12}$$

We schedule the transmission instant of f_i with a period of $f_i.sched_p$, ensuring that flow f_i can still achieve deterministic transmission with its own period $f_i.period$. In the async-to-sync mechanism and the latency adaptation mechanism, the wait time $f_i.wait$ at the first switch v_1 and the hold time $f_i.hold$ at the last switch v_{n-1} satisfy the following relationship:

$$f_i.wait + f_i.hold = f_i.sched_p. \tag{13}$$

According to (10) and (13), the E2E delay of f_i is then obtained as:

$$\begin{aligned} f_i.e2eDelay &= (f_i^{[v_{n-1},v_n]}.offset - f_i^{[v_1,v_2]}.offset) \\ &\quad + f_i^{[v_0,v_1]}.ld + f_i^{[v_{n-1},v_n]}.ld + f_i.sched_p. \end{aligned} \tag{14}$$

With the scheduling period $f_i.sched_p$ designed to be smaller than the flow period $f_i.period$, the scheduling scheme is able to ensure that the E2E delay of f_i satisfies its deadline.

E. Conflict Resolution

In a time-synchronized environment, the conflict-free constraint in the scheduling algorithm ensures that TT flows do not conflict with each other during transmission [5]. However, AES flows introduce arrival-time uncertainty, which leads to variations in $f_i.wait$ and consequently in the holding time $f_i.hold$ at the last switch v_{n-1}. For any two AES flows f_i and f_j, a collision may occur at the egress hop if their actual transmission intervals overlap after latency adaptation. Let the actual transmission start instant of f_i be $\phi_i = f_i^{[v_{n-1},v_n]}.offset_k + f_i.hold$, and that of f_j be $\phi_j = f_j^{[v_{n-1},v_n]}.offset_l + f_j.hold$. A collision occurs if their transmission intervals overlap:

$$[\phi_i,\ \phi_i + f_i^{[v_{n-1},v_n]}.ld] \cap [\phi_j,\ \phi_j + f_j^{[v_{n-1},v_n]}.ld] \neq \varnothing. \tag{15}$$

Fig. 5 illustrates a scenario where AES flows conflict with other TT flows at the last hop $[v_{n-1}, v_n]$ at the egress switch v_{n-1}. To address the above conflicts, we propose a conflict resolution mechanism based on a safety distance. AES flow f_i, sent through the latency adaptation mechanism, will not

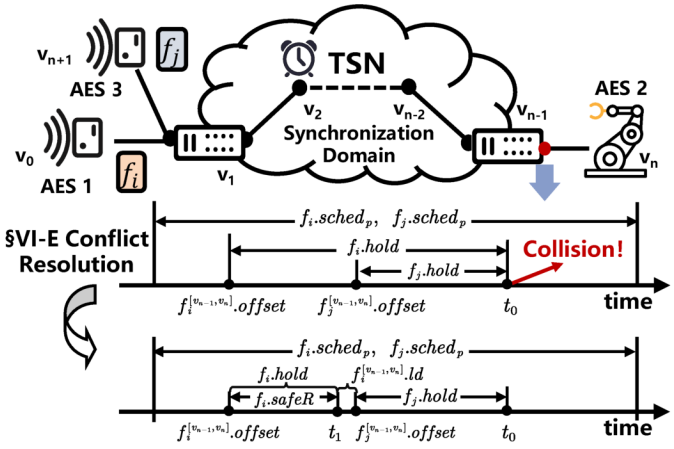


Fig. 5. Timing diagram illustrating the occurrence of collisions among AES flows and the resolution of conflicts using the conflict resolution mechanism.

conflict with other TT flows if it maintains a safety distance $f_i.safeR$ after its transmission instant. As shown in Fig. 5, the safety distance $f_i.safeR$ for f_i is defined as the minimum gap between any transmission instant $f_i^{[v_{n-1},v_n]}.offset_k$ of f_i and any subsequent transmission instant $f_j^{[v_{n-1},v_n]}.offset_m$ of other TT flows on the hop $[v_{n-1}, v_n]$, minus the link delay. Thus, $f_i.safeR$ can be expressed as:

$$f_i.safeR = \min_{f_j \in F^{[v_{n-1},v_n]}} gap_{i,j} - f_i^{[v_{n-1},v_n]}.ld. \tag{16}$$

The specific algorithm for calculating $f_i.safeR$ is described in Sec. V-C. It compares the holding time $f_i.hold$ with $f_i.safeR$, and determines the actual hold time for f_i at switch v_{n-1}.

- When $f_i.hold$ is less than $f_i.safeR$, f_i will still hold for the entire duration of $f_i.hold$ at the last switch v_{n-1}.
- When $f_i.hold$ exceeds $f_i.safeR$, f_i will hold only for the duration of the safety distance $f_i.safeR$, rather than the entire holding time.

By using the conflict resolution mechanism, the delay increase caused by conflicts between AES flows and other TT flows can be resolved.

The following analyzes the E2E delay of AES flows when the conflict resolution mechanism is introduced. When the hold time is less than the safe distance, the entire hold time is maintained, and the E2E delay remains as in (14). However, when the hold time exceeds the safe distance, only the safe distance is maintained instead of the full hold time, resulting in an earlier release. In such a case, the E2E delay becomes:

$$\begin{aligned} f_i.e2eDelay = &(f_i^{[v_{n-1},v_n]}.offset - f_i^{[v_1,v_2]}.offset) \\ &+ f_i^{[v_0,v_1]}.ld + f_i^{[v_{n-1},v_n]}.ld \\ &+ f_i.sched_p - f_i.hold + f_i.safeR. \end{aligned} \tag{17}$$

Since the hold time $f_i.hold$ is greater than the safe distance $f_i.safeR$, $f_i.sched_p - f_i.hold + f_i.safeR$ is less than $f_i.sched_p$. Therefore, after applying the conflict resolution mechanism, the E2E delay of the AES flow is less than that of the scenario without conflict, meaning no additional E2E delay and hence meeting the real-time requirements of AES flows.

According to (17), as the hold time $f_i.hold$ increases, the E2E delay decreases. When $f_i.hold$ reaches its maximum of $f_i.sched_p$, the E2E delay is minimized:

$$\begin{aligned} f_i.best = &(f_i^{[v_{n-1},v_n]}.offset - f_i^{[v_1,v_2]}.offset) \\ &+ f_i^{[v_0,v_1]}.ld + f_i^{[v_{n-1},v_n]}.ld + f_i.safeR. \end{aligned} \tag{18}$$

Conversely, when $f_i.hold$ is less than $f_i.safeR$, the E2E delay is maximized:

$$\begin{aligned} f_i.worst = &(f_i^{[v_{n-1},v_n]}.offset - f_i^{[v_1,v_2]}.offset) \\ &+ f_i^{[v_0,v_1]}.ld + f_i^{[v_{n-1},v_n]}.ld + f_i.sched_p. \end{aligned} \tag{19}$$

Using (18) and (19), we can derive the jitter for the AES flow after applying the conflict resolution mechanism:

$$f_i.jitter = f_i.sched_p - f_i.safeR. \tag{20}$$

The proposed conflict resolution mechanism does not introduce additional E2E delay, thus satisfying the real-time requirements of AES flows. Moreover, it reduces jitter to $f_i.sched_p - f_i.safeR$, enhancing the determinism of AES flow transmission.

V. Asynchronous Deterministic Access Scheduling Model

The implementation of ADA mechanism relies on ADA's scheduling model. The scheduling model is essential, specifying the transmission instant of each AES flow at each hop along its path, its scheduling period, and the safety distance between AES flows. This allows switches to reserve the necessary time slots in advance and forward frames promptly without collision.

A. Asynchronous Time-Triggered Scheduling Model

We formulate the scheduling problem as a Satisfiability Modulo Theory (SMT) problem and solve it using an SMT solver. For each AES flow f_i, we define $f_i.offsets$ and $f_i.sched_p$ as scheduling variables. In what follows, we outline common constraints for TSN scheduling in scenarios involving AES access.

1) Path Dependency Constraints: For any AES flow f_i, when transmitting frame over any two adjacent links $[v_x, v_j]$, $[v_j, v_y]$ in $f_i.route$, it is always received by the device v_j before it is transmitted by device v_j, which is formalized as:

$$\begin{aligned} &\forall f_i \in F : \forall [v_x, v_j], [v_j, v_y] \in f_i.route : \\ &f_i^{[v_j,v_y]}.offset - f_i^{[v_x,v_j]}.offset \geq f_i^{[v_j,v_y]}.ld. \end{aligned} \tag{21}$$

2) End-to-end Deadline Constraints: The E2E delay of AES flow f_i must be shorter than the application-specific deadline, which is formalized as:

$$\begin{aligned} &\forall f_i \in F : \\ &(f_i^{[v_{n-1},v_n]}.offset - f_i^{[v_1,v_2]}.offset) \\ &\quad + f_i^{[v_0,v_1]}.ld + f_i^{[v_{n-1},v_n]}.ld + f_i.sched_p \leq f_i.deadline \end{aligned} \tag{22}$$

where $[v_0, v_1]$ is the first hop, $[v_{n-1}, v_n]$ is the last hop in $f_i.route$. v_1 and v_{n-1} are the edge switches in the synchronized domain.

3) Conflict-free Constraints: An AES device or switch forwards the next frame only after the previous frame has been processed. That is, for any two TT flows f_i and f_j on link $[v_k, v_l]$, their reserved transmission time instances, repeating at their respective scheduling interval/period, must not overlap. This requirement is formalized as:

$$\begin{aligned}
&\forall [v_k, v_l] \in E, \forall f_i, f_j \in F, \\
&\forall a \in \left[0 \ldots \left(\frac{LCM\left(f_i.sched_p, f_j.sched_p\right)}{f_i.sched_p} - 1\right)\right], \\
&\forall b \in \left[0 \ldots \left(\frac{LCM\left(f_i.sched_p, f_j.sched_p\right)}{f_j.sched_p} - 1\right)\right] : \\
&\left((f_i \neq f_j) \wedge \exists {f_i}^{[v_k,v_l]} \wedge \exists {f_j}^{[v_k,v_l]}\right) \Rightarrow \\
&\left(\begin{array}{l} a \times f_i.sched_p + {f_i}^{[v_k,v_l]}.offset \geq \\ b \times f_j.sched_p + {f_j}^{[v_k,v_l]}.offset + {f_j}^{[v_k,v_l]}.ld \end{array}\right) \vee \\
&\left(\begin{array}{l} b \times f_j.sched_p + {f_j}^{[v_k,v_l]}.offset \geq \\ a \times f_i.sched_p + {f_i}^{[v_k,v_l]}.offset + {f_i}^{[v_k,v_l]}.ld \end{array}\right) \qquad (23)
\end{aligned}$$

where $LCM(f_i.sched_p, f_j.sched_p)$ is the least common multiple of scheduling periods of f_i and f_j.

4) Scheduling Period Constraints: For any AES flow f_i, the scheduling period $f_i.sched_p$ should be a factor of the period of f_i, which is formalized as:

$$\begin{aligned}
&\forall f_i \in F, \exists c \in Z^* : \\
&c \times f_i.sched_p = f_i.period. \qquad (24)
\end{aligned}$$

5) Optimization Objective: To ensure asynchronous deterministic transmission while minimizing the E2E delay for TT flows, the objective can be expressed as:

$$minimize \left(\sum_{f_i \in F} f_i.e2eDelay\right). \qquad (25)$$

B. Minimum Scheduling Period

Since each flow's scheduling period $f_i.sched_p$ is uncertain prior to scheduling, the conflict-free constraints (23) cannot be solved using integer linear programming. To address this problem, we propose a minimum scheduling period, $comCycle$. The minimum scheduling period is determined by the network size and the periodic attributes of TT flows. All TT flows' periods and scheduling periods are integer multiples of this minimum scheduling period such that:

$$\begin{aligned}
&\forall f_i \in F, \exists d \in Z^* : \\
&d \times comCycle = f_i.sched_p. \qquad (26)
\end{aligned}$$

Based on the minimum scheduling period, we reformulated the conflict-free constraints (23). That is, the reserved transmission instances of TT flows f_i and f_j on link $[v_k, v_l]$, repeating with the minimum scheduling period, must not overlap. This constraint is formalized as:

$$\begin{aligned}
&\forall [v_k, v_l] \in E, \forall f_i, f_j \in F, \\
&\forall a \in \left[0 \ldots \left(\frac{LCM\left(f_i.period, f_j.period\right)}{comCycle} - 1\right)\right], \\
&\forall b \in \left[0 \ldots \left(\frac{LCM\left(f_i.period, f_j.period\right)}{comCycle} - 1\right)\right] : \\
&\left((f_i \neq f_j) \wedge \exists {f_i}^{[v_k,v_l]} \wedge \exists {f_j}^{[v_k,v_l]}\right) \Rightarrow \\
&\left(\begin{array}{l} a \times comCycle + {f_i}^{[v_k,v_l]}.offset \geq \\ b \times comCycle + {f_j}^{[v_k,v_l]}.offset + {f_j}^{[v_k,v_l]}.ld \end{array}\right) \vee \\
&\left(\begin{array}{l} b \times comCycle + {f_j}^{[v_k,v_l]}.offset \geq \\ a \times comCycle + {f_i}^{[v_k,v_l]}.offset + {f_i}^{[v_k,v_l]}.ld \end{array}\right). \qquad (27)
\end{aligned}$$

This ensures that there are no conflicts between any two TT flows.

C. Calculation of Safety Distance

The scheduling algorithm ensures deterministic transmission for AES flows within the synchronized domain. However, the uncertainty of asynchronous TT arrival times in the synchronized domain causes $f_i.wait$ to be uncertain, which in turn makes $f_i.hold$ uncertain. This may result in conflicts with other TT flows at the last switch on the final hop. The conflict resolution mechanism in Sec. IV-E ensures that AES flows maintain a safe time interval from other TT flows at the last hop, preventing transmission conflicts with other TT flows.

For any hop $[v_{n-1}, v_n]$, where v_{n-1} is the last switch and v_n is an AES device, let the set of AES flows passing through this hop be denoted as $F^{[v_{n-1},v_n]}$. The safe interval on $[v_{n-1}, v_n]$ is calculated as shown in Alg. 1 which consists of the following steps:

1) Iterate through the set of TT flows on $[v_{n-1}, v_n]$. For each flow f_i, initialize its safety interval as its scheduling period $f_i.sched_p$.
2) For each flow f_j that may cause a transmission conflict with f_i, initialize the minimum offset gap $offsetGap$ between f_i and f_j as $f_i.sched_p$.
3) Calculate each transmission instant q of f_j within the least common multiple of scheduling period of f_i and f_j. If q occurs after $f_i.offset$, compute the interval (mod $f_i.sched_p$) between q and $f_i.offset$, and update $offsetGap$ with the minimum interval.
4) After calculating the intervals for all conflicting TT flows, update the safety distance accordingly.

The ADA scheduling model in Sec. V provides the required transmission instants, scheduling periods, and safety distances for achieving ADA mechanism. This information is then converted to a scheduling table and configured on the switches. The ADA mechanism in Sec. IV supports deterministic access for AES and ensures deterministic transmission for AES flows.

VI. Evaluation

We conduct two parts of the evaluation. One is the algorithmic evaluation of ADA including Sec. VI-B–VI-D. The other is the deployment evaluation with the real hardware implementation of ADA in Sec. VI-E.

Algorithm 1 Safety Distance Calculation

Input: Flow Set on Hop $[v_{n-1}, v_n]$: $F^{[v_{n-1},v_n]}$
Output: Safety Distance Set of $F^{[v_{n-1},v_n]}$

foreach $f_i \in F^{[v_{n-1},v_n]}$ **do**
  $minGap_i = f_i.sched_p$
  foreach $f_j \in F^{[v_{n-1},v_n]} \wedge i \neq j$ **do**
    $offsetGap = f_i.sched_p$
    foreach $k \in [0 \ldots (\frac{LCM(f_i.sched_p, f_j.sched_p)}{f_j.sched_p} - 1)]$ **do**
      $q = (k \times f_j.sched_p + f_j^{[v_{n-1},v_n]}.offset) - f_i^{[v_{n-1},v_n]}.offset$
      if $q \geq 0$ **then**
        $m = q \mod f_i.sched_p$
        $offsetGap = min(offsetGap, m)$
    $f_i.safeR = min(f_i.safeR, offsetGap - f_i^{[v_{n-1},v_n]}.ld)$
return $\{f_i.safeR | \forall f_i \in F^{[v_{n-1},v_n]}\}$

A. Evaluation Setup

1) Implementation: We implement the ADA scheduling model in Java and solve it using the CPLEX integer linear programming solver [37]. All scheduling algorithms are executed on a laptop equipped with an Intel i7-1165G7 CPU and 32 GB of RAM. The rates of both the switch port and the link are configured to be 1 Gbps. We prototype ADA on real TSN hardware. Specifically, we designed an 8-port Gigabit TSN switch compliant with IEEE TSN standards [4], [38], implemented on a Xilinx Virtex-7 XC7VX485T FPGA. The switch is synthesized using Vivado 2016.1 and operates at a 125 MHz clock frequency. AES devices are emulated using a standard test instrument, IXIA [39], which can periodically transmit and receive data with nanosecond-level timing accuracy.

2) Delay Model: The E2E delay of a flow is composed of the delays accumulated over all the hops along its transmission path. Each hop delay of a flow——measured from the transmission start instant of a frame at one device to the transmission start instant at the next device——consists of five components: processing delay T_P, propagation delay T_{Pr}, receive delay T_R, forwarding delay T_F, and queueing delay T_Q. Fig. 6 illustrates the frame processing procedure between two consecutive transmission offsets and the composition of the corresponding delay components, defined as follows.

- Processing delay (T_P): the time from frame fetching to transmission of the first bit on the output port, which is set to $1\mu s$ based on our implemented TSN switch.
- Propagation delay (T_{Pr}): the physical signal propagation time between two directly connected devices. It is calculated as the ratio of the physical distance to the signal propagation speed over the transmission medium. The propagation delay is set to $50ns$, which corresponds to a typical 10-meter Ethernet cable.

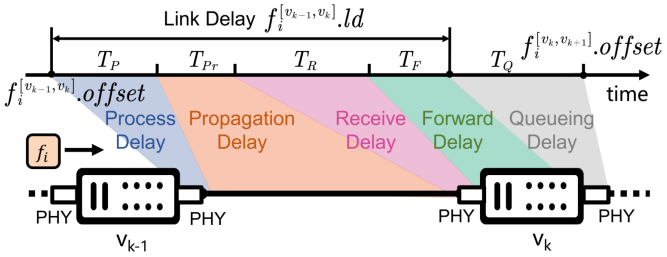


Fig. 6. Illustration of the per-hop delay composition of a flow.

- Receive delay (T_R): the time for fully receiving a frame into a per-port frame pool, calculated as frame size divided by port rate. For a maximum-size frame at 1 Gbps, this is approximately $12\mu s$.
- Forwarding delay (T_F): the time consumed after a switch fully receives a frame, including frame parsing, forwarding table lookup, and internal switching operations that move the abstract (including flow ID, frame length, and address in the frame pool) of a frame to the TT queue of the corresponding output port. This delay is set to $800ns$ based on measurements on our implemented TSN switch.
- Queueing delay (T_Q): the waiting time in the egress TT queue until the scheduled transmission instant. For a given flow instance, it is computed as the time interval between the moment when the frame is completely forwarded into the output port's TT queue and its scheduled transmission instant.

Except for the queueing delay, the per-hop components are primarily determined by hardware implementation and frame size, and are typically on the microsecond-to-nanosecond scale. In the scheduling model, these components are treated as the parameters of a link delay for each hop $[v_{k-1}, v_k]$, denoted as:

$$f_i^{[v_{k-1},v_k]}.ld = T_P + T_{Pr} + T_R + T_F. \tag{28}$$

The queueing delay, which depends on the assigned transmission offset at each hop, is the dominant contributor and the main factor differentiating the compared algorithms. In ADA, the queueing delay is determined by the conflict-free transmission offsets assigned to each flow. For a flow f_i transmitted along nodes v_{k-1}, v_k, and v_{k+1}, the queueing delay at hop $[v_k, v_{k+1}]$ can be computed as:

$$T_Q = f_i^{[v_k,v_{k+1}]}.offset - \left(f_i^{[v_{k-1},v_k]}.offset + f_i^{[v_{k-1},v_k]}.ld\right). \tag{29}$$

Once the transmission offsets are determined, the corresponding queueing delays are fixed within the TSN domain.

3) Network Topology: For the feasibility evaluation, we selected four typical industrial network topologies inspired by industrial and automotive application requirements [32]. As shown in Fig. 7, the network topologies include a small ring mesh (SRM), a medium ring (MR), a medium mesh (MM), and a medium tree with depth 2 (MT). For the applicability evaluation, we selected the realistic Orion Crew Exploration Vehicle (CEV) topology [40], as shown in Fig. 8. This network consists of 13 switches and 31 end systems, 44 devices in total.

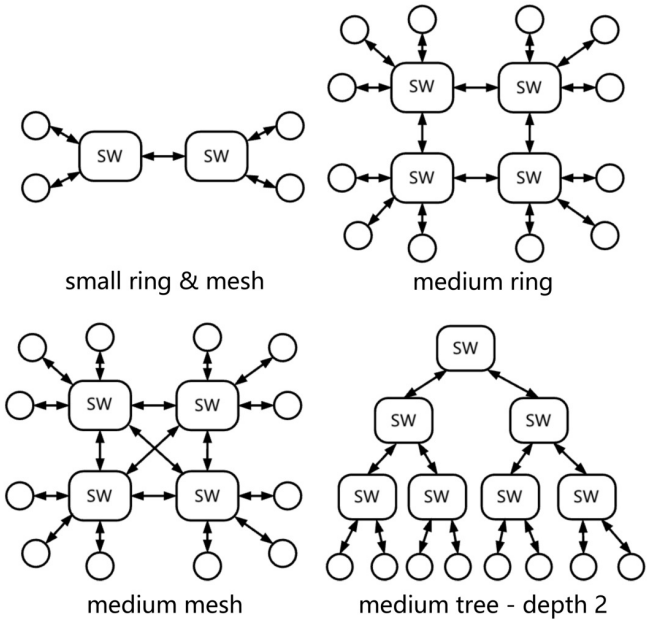


Fig. 7. Network topologies used in feasibility evaluation.

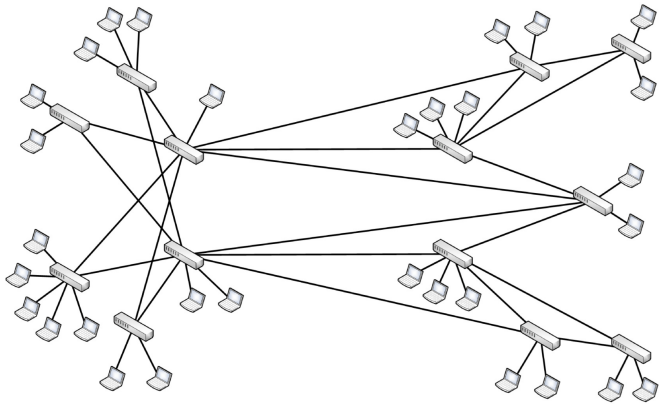


Fig. 8. Topology of the Orion CEV network used in applicability evaluation.

In this setup, all end systems are AES devices, and all switches are time-synchronized.

4) Flow Requirement: All TT flows within our experiment are randomly generated, adhering to the IEC/IEEE 60802 standard [41] for industrial automation networks. For each TT flow, a source end system is randomly selected from the set of all end systems within the network topology. A destination end system is then selected randomly from the remaining end systems, excluding the source. Frame lengths are randomly assigned between 64 and 1518 bytes. Flow periods are randomly selected from $\{1.5ms, 2.5ms, 3.5ms, 5ms, 7.5ms, 10ms\}$. All TT traffic follows a constant bit rate model, and there is no randomness in the frame generation periods. The E2E delay deadline for each flow is the same as its period. Flows' routes are determined by a shortest path algorithm [42] before scheduling.

5) Comparative Methods: We compare the performance of ADA with the following prior methods:

TABLE I
DETAILS OF THE SETUP USED IN THE FEASIBILITY EVALUATION

No.	Network Topology	Total No. of SWs	Total No. of ESs	Total No. of Flows	Hyperperiod (ms)
1	SRM	2	3	9	15
2	SRM	3	3	11	70
3	SRM	3	4	15	70
4	MR	4	6	15	30
5	MR	4	8	21	210
6	MR	5	11	27	210
7	MM	4	5	13	15
8	MM	6	12	30	210
9	MM	7	13	35	210
10	MT	7	8	18	105
11	MT	7	8	25	105
12	MT	7	12	32	210

AsyncTT [5]: A synchronized method that ignores asynchronous transmission characteristics and uses a synchronous time-triggered transmission mechanism for AES flows. As a TT-based mechanism, its queueing delay is computed in the same way as in ADA.

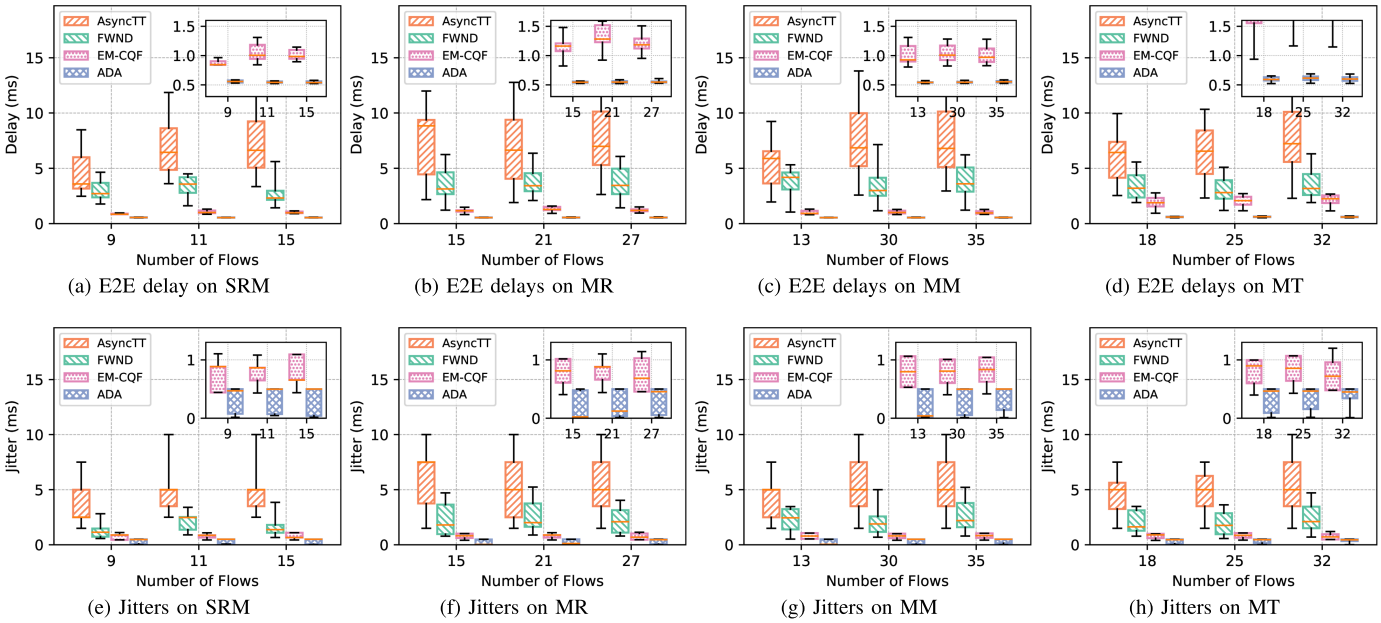
FWND [10]: A flexible, window-based TSN scheduling algorithm that performs a worst-case delay analysis and supports heterogeneous TSN networks with unsynchronized end systems. In FWND, the queueing delay is derived from the worst-case analysis using network calculus. Specifically, it is computed based on the backlog bound at each output port, which depends on the cumulative arrival and service curves of the flows sharing the same link. The resulting delay represents the upper bound on the waiting time before a frame can be transmitted.

EM-CQF [34]: An enhanced Multi-CQF scheme based on the CQF mechanism. The number of cyclic queues is configured as 3. In EM-CQF, the queueing delay is determined by the queueing offset assigned within the cyclic queueing structure and the duration of each cycle. Each frame must wait until its designated transmission cycle becomes active, causing the queueing delay to vary with the configured cycle length and the frame's assigned offset within the CQF rotation.

B. Feasibility Evaluation

Table I presents detailed information on the test cases used in the feasibility evaluation experiment. The minimum scheduling period for ADA is set to $500\mu s$. Fig. 9 presents an overview of the performance of ADA compared with AsyncTT, FWND, and EM-CQF across various typical network topologies using box plots. Each box illustrates the distribution of the E2E delay or jitter among all flows under a given method. The central line within each box denotes the median value, the box boundaries represent the first and third quartiles, and the whiskers extend to the minimum and maximum values within $1.5\times$ the inter-quartile range. Overall, ADA demonstrates the lowest average E2E delay and average jitter, outperforming its counterparts.

Figs. 9(a–d) show the delay results. All the delay results reported in this section correspond to the worst-case E2E delays of the evaluated flows. Across different network



(a) E2E delay on SRM (b) E2E delays on MR (c) E2E delays on MM (d) E2E delays on MT

(e) Jitters on SRM (f) Jitters on MR (g) Jitters on MM (h) Jitters on MT

Fig. 9. (a)–(d) present the E2E delay performance of ADA compared to other methods across various network topologies, while (e)–(h) depict the corresponding jitter results.

topologies, including SRM, MR, MM, and MT, ADA consistently outperforms other methods. Compared to the traditional AsyncTT method, ADA achieves an average E2E delay of just 8.38% of that of AsyncTT, reducing the delay from 6.71 ms to 0.56 ms, an improvement of approximately 91.6%. Additionally, ADA reduces average E2E delay by 83.5% and 57.1% over FWND and EM-CQF, respectively. Figs. 9(e–h) show the jitter evaluation results. Across four different network topologies, ADA consistently exhibits superior jitter control to other methods. Specifically, ADA reduces average jitter by 93.8% over the traditional AsyncTT method, bringing it down from 4.98 ms to just 0.31 ms. Additionally, ADA reduces average jitter by 85.4% and 60.7% over FWND and EM-CQF, respectively.

The results arise primarily from differences in their scheduling mechanisms and offset assignments, which directly determine the queueing delay at each hop. In AsyncTT, the lack of synchronization between AES devices and the TSN domain causes AES frames to arrive at different instants within each period. Consequently, the queueing delay at the ingress switch varies within a period, leading to fluctuations in both E2E delay and jitter. FWND allows flows to contend for transmission within flexible windows on each hop. The non-deterministic transmission instants introduce variable queueing delays, resulting in larger variations in E2E delay and jitter. EM-CQF, as an enhancement of the CQF mechanism, still exhibits variability at the final hop, where the transmission can occur anywhere within a cycle, causing residual queueing delay variation. In contrast, ADA introduces deterministic wait and hold times that align transmission offsets across hops, thereby minimizing queueing delay fluctuations. By utilizing the scheduling period to minimize waiting offsets, ADA effectively mitigates the asynchronous access issue. Furthermore, ADA's conflict resolution mechanism further reduces E2E delay by allowing safe early transmissions when potential conflicts are detected at the last hop connected to the receiving end device.

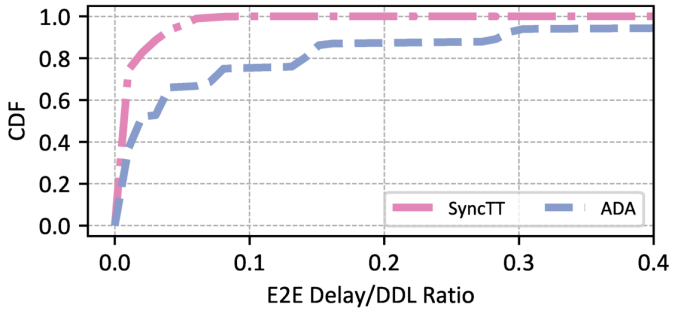
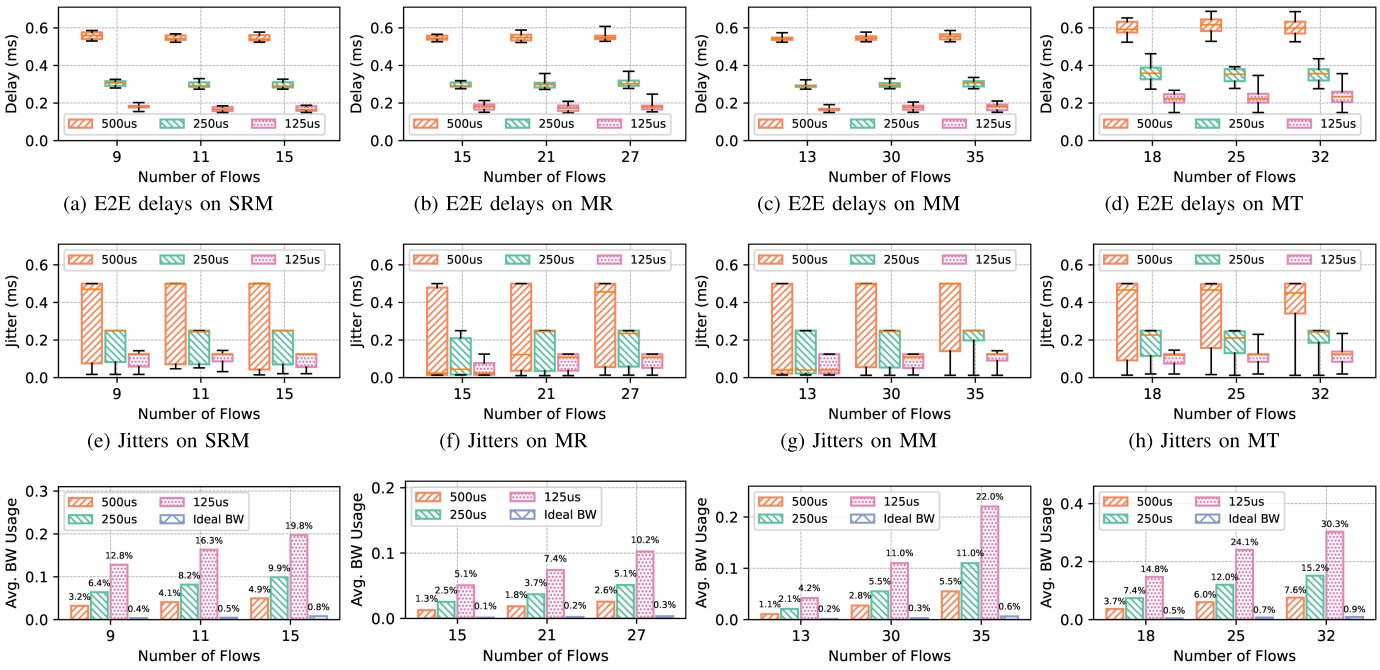


Fig. 10. The CDF result comparison of applicability evaluation.

C. Applicability Evaluation

We now evaluate the applicability of ADA in large, complex networks (CEV). Initially, 100 TT flows are randomly generated and scheduled using both synchronous TT transmission (SyncTT) and ADA. We compare the E2E delays of TT flows scheduled with SyncTT in synchronized networks with those scheduled with ADA in AES devices to demonstrate the delay performance of ADA approaching that of SyncTT.

Fig. 10 illustrates the cumulative distribution function (CDF) of the Delay/DDL Ratio, which is the ratio of the actual E2E delay (Delay) to the deadline (DDL) of flows. The CDF shows that using ADA results in a slightly higher latency than direct time-triggered scheduling in synchronized systems. This increase stems from ADA's introduction of a scheduling period to ensure deterministic transmission of AES flows, incurring an additional delay equal to the scheduling



(a) E2E delays on SRM (b) E2E delays on MR (c) E2E delays on MM (d) E2E delays on MT

(e) Jitters on SRM (f) Jitters on MR (g) Jitters on MM (h) Jitters on MT

(i) Average bandwidth usage on SRM (j) Average bandwidth usage on MR (k) Average bandwidth usage on MM (l) Average bandwidth usage on MT

Fig. 11. Performance of ADA for different minimum scheduling periods across four network topologies. (a)–(d) show the average E2E delays, (e)–(h) present the average jitters, and (i)–(l) depict the average bandwidth usage for the SRM, MR, MM, and MT topologies, respectively.

period compared to synchronous flows. However, the latency increase remains within an acceptable range. The vast majority of TT flows achieve the E2E latency within a Delay/DDL Ratio of 30%, and the overall latency under ADA remains within the maximum latency constraint, not exceeding 50% of the allowed limit.

D. Impact of the Minimum Scheduling Period

We now evaluate the performance of ADA under different minimum scheduling periods ($comCycle$) in terms of E2E latency, jitter, and average bandwidth usage, based on the test cases summarized in Table I. The average bandwidth usage is defined as the mean ratio of the total reserved bandwidth of all scheduled flows to the link capacity across all data links, as given by

$$\bar{U} = \frac{1}{C \cdot |E|} \sum_{[v_a, v_b] \in E} \sum_{f_i \in F^{[v_a, v_b]}} \frac{f_i.length}{f_i.sched_p} \tag{30}$$

where C denotes the link capacity and $|E|$ is the number of data links.

We consider three minimum scheduling periods: $500\mu s$, $250\mu s$, and $125\mu s$. Figs. 11(a–d) show the E2E latency results. For $comCycle = 500\mu s$, $250\mu s$, and $125\mu s$, the average E2E latencies across the test cases are $0.56ms$, $0.31ms$, and $0.18ms$, respectively. Figs. 11(e–h) show the average jitters under these $comCycle$s equal to $0.31ms$, $0.16ms$, and $0.09ms$, respectively. Figs. 11(i–l) show the average bandwidth utilization across all links in the network topology for the respective ADA scheduling results. These are compared against the actual bandwidth requirement of AES flows, denoted as "Ideal BW" in the figure. Compared to the actual bandwidth requirement, ADA introduces 3.25%, 6.96%, and 14.38% bandwidth overheads with $comCycle = 500\mu s$, $250\mu s$, and $125\mu s$, respectively.

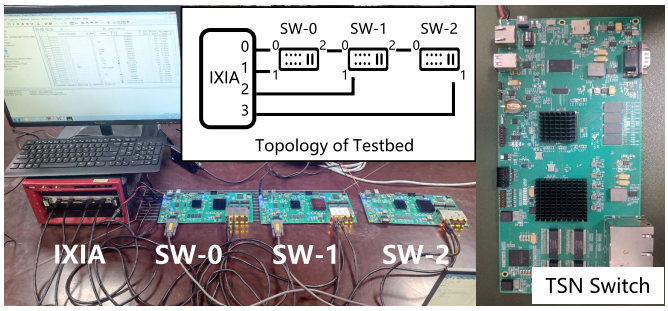
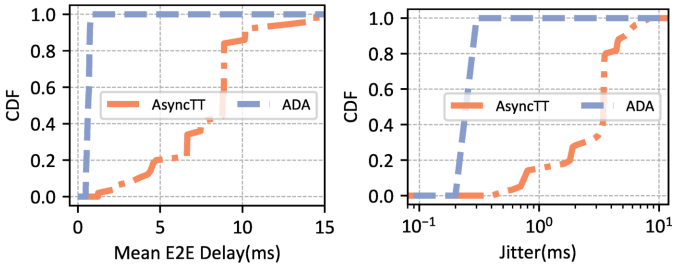


Fig. 12. The testbed for validating the asynchronous deterministic access mechanism.

These results confirm that reducing the minimum scheduling period increases bandwidth consumption, because finer-grained time slots result in more frequent reservation for each flow instance, hence increasing the total reserved bandwidth. However, this additional bandwidth reservation brings significant benefits, including a notable reduction in the E2E delay and jitter of AES flows. This demonstrates a clear trade-off between bandwidth utilization and transmission determinism. When there are sufficient bandwidth resources, this trade-off provides a practical and effective vehicle for providing deterministic transmission.

E. Testbed Deployment

To corroborate ADA's ability to ensure deterministic and real-time transmission for AES flows in practical AES devices'



(a) CDF of E2E delay. (b) CDF of jitter.

Fig. 13. The CDF result comparison on the testbed.

access scenarios, we built a testbed with 3 switches and a standard tester, as shown in Fig. 12. The IXIA is a standard tester with four ports, used as AES devices for sending and receiving AES flows. SW-0, SW-1, and SW-2 are TSN switches equipped with our ADA mechanism.

In each experiment, we randomly generate and schedule 50 flows using the ADA scheduling model, and then configure the scheduling results on each switch. We record transmission delays with a nanosecond-level precision. The average E2E delay and jitter for all flows are measured over 1,000 periods. Fig. 13 shows the CDFs of testbed results across 10 randomly generated experiments, where Fig. 13a provides the CDF of the average latency for all TT flows. Under ADA the average latency is reduced from 7.9 ms (using AsyncTT) to 0.51 ms. Fig. 13b shows the CDF of the jitter for all TT flows. Using ADA reduces jitter by 92.1%, from an average of 3.2 ms (with AsyncTT) to 0.25 ms, with a maximum jitter under ADA not exceeding 0.3 ms. The increased latency and jitter under AsyncTT occur because AES flows miss transmission offsets and wait until the next transmission offset, causing delays and jitter. ADA effectively addresses all of these issues. Overall, ADA ensures real-time, deterministic transmission for AES flows in AES access scenarios.

VII. Conclusion

We have presented ADA as an effective way of achieving deterministic transmission of AES flows in TSN. ADA enhances real-time performance and reduces jitter, significantly improving the determinism of flows in AES devices. Our experimental evaluations corroborate ADA's capability of significantly lowering both the E2E delay and jitter compared to SOTA methods, thus enhancing determinism in TSN with AES devices.

References

[1] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the Internet of Things and industry 4.0," *IEEE Ind. Electron. Mag.*, vol. 11, no. 1, pp. 17–27, Mar. 2017.

[2] L. Lo Bello and W. Steiner, "A perspective on IEEE time-sensitive networking for industrial communication and automation systems," *Proc. IEEE*, vol. 107, no. 6, pp. 1094–1120, Jun. 2019.

[3] D. Bruckner et al., "An introduction to OPC UA TSN for industrial communication systems," *Proc. IEEE*, vol. 107, no. 6, pp. 1121–1131, Jun. 2019.

[4] IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks–Amendment 25: Enhancements for Scheduled Traffic, IEEE Standard 802.1Qcd, 2016, pp. 1–57.

[5] W. Steiner, "An evaluation of SMT-based schedule synthesis for timetriggered multi-hop networks," in *Proc. 31st IEEE Real-Time Syst. Symp.*, Nov. 2010, pp. 375–384.

[6] Z. Zhao et al., "Access mechanism for period flows of non-deterministic end systems for time-sensitive networks," *Comput. Netw.*, vol. 231, Jul. 2023, Art. no. 109805. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128623002505

[7] IEEE Standard for Local and Metropolitan Area Networks–Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-sensitive Streams, IEEE Standard 802.1Qav, 2010.

[8] IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks–Amendment 34: Asynchronous Traffic Shaping, IEEE Standard 802.1Qcc, 2020, pp. 1–151.

[9] N. Reusch, L. Zhao, S. S. Craciunas, and P. Pop, "Window-based schedule synthesis for industrial IEEE 802.1 Qbv TSN networks," in *Proc. 16th IEEE Int. Conf. Factory Commun. Syst. (WFCS)*, Apr. 2020, pp. 1–4.

[10] M. Barzegaran, N. Reusch, L. Zhao, S. S. Craciunas, and P. Pop, "Realtime traffic guarantees in heterogeneous time-sensitive networks," in *Proc. 30th Int. Conf. Real-Time Netw. Syst.*, 2022, pp. 46–57.

[11] N. Reusch, M. Barzegaran, L. Zhao, S. S. Craciunas, and P. Pop, "Configuration optimization for heterogeneous time-sensitive networks," *Real-Time Syst.*, vol. 59, no. 4, pp. 705–747, 2023. [Online]. Available: https://link.springer.com/article/10.1007/s11241-023-09414-0

[12] Z. Li et al., "Time-triggered switch-memory-switch architecture for time-sensitive networking switches," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 1, pp. 185–198, Jan. 2020.

[13] C. Li, Z. Li, T. Li, C. Li, and B. Wang, "A deterministic embedded end-system tightly coupled with TSN schedule," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 42, no. 11, pp. 3707–3719, Nov. 2023.

[14] R. Serna Oliver, S. S. Craciunas, and W. Steiner, "IEEE 802.1Qbv gate control list synthesis using array theory encoding," in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS)*, Apr. 2018, pp. 13–24.

[15] N. G. Nayak, F. Dürr, and K. Rothermel, "Incremental flow scheduling and routing in time-sensitive software-defined networks," *IEEE Trans. Ind. Informat.*, vol. 14, no. 5, pp. 2066–2075, May 2018.

[16] N. Wang, Q. Yu, H. Wan, X. Song, and X. Zhao, "Adaptive scheduling for multicluster time-triggered train communication networks," *IEEE Trans. Ind. Informat.*, vol. 15, no. 2, pp. 1120–1130, Feb. 2019.

[17] J. Falk, F. Dürr, and K. Rothermel, "Time-triggered traffic planning for data networks with conflict graphs," in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS)*, Apr. 2020, pp. 124–136.

[18] Q. Yu and M. Gu, "Adaptive group routing and scheduling in multicast time-sensitive networks," *IEEE Access*, vol. 8, pp. 37855–37865, 2020.

[19] Y. Zhou, S. Samii, P. Eles, and Z. Peng, "Reliability-aware scheduling and routing for messages in time-sensitive networking," *ACM Trans. Embedded Comput. Syst.*, vol. 20, no. 5, pp. 1–24, Sep. 2021.

[20] M. Vlk, Z. Hanzálek, and S. Tang, "Constraint programming approaches to joint routing and scheduling in time-sensitive networks," *Comput. & Ind. Eng.*, vol. 157, Jun. 2021, Art. no. 107317. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835221002217

[21] G. Patti, L. L. Bello, and L. Leonardi, "Deadline-aware online scheduling of TSN flows for automotive applications," *IEEE Trans. Ind. Informat.*, vol. 19, no. 4, pp. 5774–5784, Apr. 2023.

[22] Y. Huang, S. Wang, X. Zhang, T. Huang, and Y. Liu, "Flexible cyclic queuing and forwarding for time-sensitive software-defined networks," *IEEE Trans. Netw. Service Manage.*, vol. 20, no. 1, pp. 533–546, Mar. 2023.

[23] IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks–Amendment 29: Cyclic Queuing and Forwarding, IEEE Standard 802.1Qcd, 2017, pp. 1–30.

[24] Z. Li et al., "A flattened-priority framework for mixed-criticality systems," *IEEE Trans. Ind. Electron.*, vol. 67, no. 11, pp. 9862–9872, Nov. 2020.

[25] Z. Li, H. Wan, Y. Deng, K. Xiong, and X. Song, "A resource-efficient priority scheduler for time-sensitive networking switches," *CCF Trans. Netw.*, vol. 3, no. 1, pp. 1–14, Sep. 2020.

[26] J. Yan et al., "TSN-builder: Enabling rapid customization of resource-efficient switches for time-sensitive networking," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, Jul. 2020, pp. 1–6.

[27] W. Fu, W. Quan, J. Yan, and Z. Sun, "Fenglin-I: An open-source time-sensitive networking chip enabling agile customization," *IEEE Trans. Comput.*, vol. 72, no. 1, pp. 140–153, Jan. 2023.

[28] L. Zhao, P. Pop, Z. Zheng, and Q. Li, "Timing analysis of AVB traffic in TSN networks using network calculus," in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS)*, Apr. 2018, pp. 25–36.
[29] L. Zhao, P. Pop, Z. Zheng, H. Daigmorte, and M. Boyer, "Latency analysis of multiple classes of AVB traffic in TSN with standard credit behavior using network calculus," *IEEE Trans. Ind. Electron.*, vol. 68, no. 10, pp. 10291–10302, Oct. 2021.
[30] Z. Zhou, Y. Yan, M. Berger, and S. Ruepp, "Analysis and modeling of asynchronous traffic shaping in time sensitive networks," in *Proc. 14th IEEE Int. Workshop Factory Commun. Syst. (WFCS)*, Jun. 2018, pp. 1–4.
[31] Z. Zhou, M. S. Berger, S. R. Ruepp, and Y. Yan, "Insight into the IEEE 802.1Qcr asynchronous traffic shaping in time sensitive network," *Adv. Sci., Technol. Eng. Syst. J.*, vol. 4, no. 1, pp. 292–301, 2019.
[32] L. Zhao, P. Pop, and S. Steinhorst, "Quantitative performance comparison of various traffic shapers in time-sensitive networking," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 3, pp. 2899–2928, Sep. 2022.
[33] J. Joung and J. Kwon, "Zero jitter for deterministic networks without time-synchronization," *IEEE Access*, vol. 9, pp. 49398–49414, 2021.
[34] D. Yang, Z. Cheng, W. Zhang, H. Zhang, and X. Shen, "Burstaware time-triggered flow scheduling with enhanced multi-CQF in time-sensitive networks," *IEEE/ACM Trans. Netw.*, vol. 31, no. 6, pp. 2809–2824, Dec. 2023.
[35] IEEE Standard for Local and Metropolitan Area Networks–Frame Replication and Elimination for Reliability, IEEE Standard 802.1CB, Sep. 2017.
[36] IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks–Amendment 28: Per-stream Filtering and Policing, IEEE Standard 802.1Qca, 2017, pp. 1–65.
[37] (2019). *IBM ILOG CPLEX Optimization Studio*. [Online]. Available: https://www.ibm.com/products/ilog-cplex-optimization-studio
[38] IEEE Standard for Local and Metropolitan Area Networks–timing and Synchronization for Time-sensitive Applications, IEEE Standard 802.1AS, 2020, pp. 1–421.
[39] (2023). *IXIA Traffic Generator Analyzer*. [Online]. Available: https://www.keysight.com/us/en/products/network-test/network-testhardware.html
[40] D. Tamas-Selicean and P. Pop, "Optimization of TTEthernet networks to support best-effort traffic," in *Proc. IEEE Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2014, pp. 1–4.
[41] Time-sensitive Networking Profile for Industrial Automation, IEEE Standard 60802, 2018.
[42] G. Gallo and S. Pallottino, "Shortest path algorithms," *Ann. Oper. Res.*, vol. 13, no. 1, pp. 1–79, 1988.

Cheng Long received the B.S. degree from the School of Computer Science and Technology, Beijing Jiaotong University, China, in 2023, where he is currently pursuing the Ph.D. degree. His research interests include time-sensitive networking and coordinated scheduling of synchronous and asynchronous traffic for real-time and deterministic communication.

Wenlin Zhu received the M.S. degree in computer science and technology from Beijing Jiaotong University, China, in 2024. He is currently with China United Network Communications Company Ltd., Linyi Branch, China. His research interests include time-sensitive networking and asynchronous deterministic transmission guarantee.

Zixiao Wang received the B.S. degree in computer science and technology from the College of Information Science and Engineering, Hohai University, China, in 2023. He is currently pursuing the M.S. degree with Beijing Jiaotong University. His research interests include industrial real-time networks, 5G, and traffic scheduling.

Yiming Zheng received the B.E. degree from the School of Electronic Information Engineering, Northeast Forestry University, China, in 2021, and the M.S. degree in wireless communication systems from the University of Sheffield, China, in 2022. He is currently pursuing the Ph.D. degree with Beijing Jiaotong University, China.

His research interests include traffic scheduling for 5G and 6G and time triggering combined with beam forming.

Zonghui Li received the B.S. degree in computer science from Beijing Information Science and Technology University in 2010, the M.S. degree from the Institute of Microelectronics, Tsinghua University, Beijing, China, in 2014, and the Ph.D. degree from the School of Software, Tsinghua University, in 2019.

He is currently an Associate Professor with the School of Computer Science and Technology, Beijing Jiaotong University, Beijing. His research interests include embedded and high performance computing and real-time embedded systems, especially for industrial control networks and fog computing.

Kang G. Shin (Life Fellow, IEEE) is currently the Kevin & Nancy O'Connor Professor Emeritus of Computer Science with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor. His current research focuses on safe and secure embedded real-time and cyber-physical systems and QoS-sensitive computing and networking. He has supervised the completion of 93 Ph.D.'s and authored/co-authored about 1000 technical articles, a textbook and more than 60 patents or invention disclosures, and received numerous awards, including 2000 and 2010 USENIX Annual Technical Conferences, the 2003 IEEE Communications Society William R. Bennett Prize Paper Award and the 1987 Outstanding IEEE Transactions of Automatic Control Paper Award, the Best Paper Awards from 2023 VehicleSec, 2011 ACM International Conference on Mobile Computing and Networking, 2011 IEEE International Conference on Autonomic Computing, 2019 Caspar Bowden Award for Outstanding Research in Privacy Enhancing Technologies, 2023 IEEE TCCPS Technical Achievement Award, 2023 SIGMOBILE Test-of-Time Award, and 2026 IEEE TC on Distributed Processing (TCDP) Award for Outstanding Technical Achievement. He has also received several institutional awards, including the Research Excellence Award in 1989, Outstanding Achievement Award in 1999, Distinguished Faculty Achievement Award in 2001, and Stephen Attwood Award in 2004 from The University of Michigan (the highest honor bestowed to Michigan Engineering faculty); a Distinguished Alumni Award of the College of Engineering, Seoul National University, in 2002; 2003 IEEE RTC Technical Achievement Award; and 2006 Ho-Am Prize in Engineering (the highest honor bestowed to Korean-origin engineers). He has chaired the Michigan Computer Science and Engineering Division for four years starting 1991 and also several major conferences, including 2009 ACM MobiCom and 2005 ACM/USENIX MobiSys. He was a co-founder of a couple of startups, licensed some of his technologies to industry, and served as an Executive Advisor for Samsung Research.