Time-Aware Stream Reservation for Distributed TSN

Ching-Chih Chuang*, Yuan-Yao Shih†, Jian-Cheng Chen‡ and Ai-Chun Pang‡§
*Department of Information Management, National Pingtung University, Taiwan
†Department of Communications Engineering, National Chung Cheng University, Taiwan
‡Graduate Institute of Networking and Multimedia, National Taiwan University, Taiwan
§Department of Computer Science and Information Engineering, National Taiwan University, Taiwan
Email: ccc@mail.nptu.edu.tw, yyshih@ccu.edu.tw, r07944031@csie.ntu.edu.tw, acpang@csie.ntu.edu.tw

***Abstract*—Recently, a novel network architecture called Time-Sensitive Networking (TSN), which supports high bandwidth and deterministic communication, emerges to satisfy the requirements of safety-critical real-time applications for industrial automation. TSN can operate in centralized and distributed models, and the distributed model is more suitable for the safety-critical applications that require high network availability and reliability. However, the existing approach for the TSN's fully distributed model makes a pessimistic latency bound estimation due to the problems of lower priority streams and dependency. The pessimistic estimation will result in a decrease in the schedulability of TSN streams. This paper identifies and evaluates the impact of the two problems via a case study and then proposes a TDMA-based stream reservation approach to alleviate the two problems. The simulation results validate the proposed approach in terms of the number of accepted streams and computation time.**

***Index Terms*—Time-sensitive networking (TSN), distributed network model, stream reservation**

I. Introduction

In recent years, global industries attempt to promote the concept of "Industry 4.0" to enable product customization and precision production for future factories. In order to realize the goals of Industry 4.0, many industrial applications need high bandwidth and deterministic capability to meet the strict network requirements (e.g., hard deadline). Hence, several advanced network protocols have been proposed, and Time-Sensitive Networking (TSN), defined by IEEE 802.1 family, is regarded as one of the promising next-generation in-factory networking protocols.

TSN is an Ethernet-based protocol with considering deterministic communication and supporting high link bandwidth (from 1Gbps to 10Gbps). To enable deterministic communication over the Ethernet, the extension, IEEE 802.1Qcc [1], is developed and depicts three network configuration models:

- **Fully Distributed Model**. An end-device (e.g., talker or listener) can directly transmit a message with user requirements (i.e., user model) to another end-device via a TSN user protocol. The network configuration (e.g., routing or scheduling) can be calculated based on the algorithm of the network protocol.
- **Centralized Network/Distributed User Model**. An additional centralized manager, called Centralized Network Configuration (CNC), is required, and both user model and network configuration should be precessed in CNC.
- **Fully Centralized Model**. This model uses an entity called Centralized User Configuration (CUC) responsible for handling the user model. Then, all user requirements will be exchanged between CNC and CUC, and CNC is responsible for configuring TSN features on each bridge via a remote network protocol.

In the Fully Centralized and Centralized Network/Distributed User Model, CNC has a complete view of the network to find near-optimal scheduling for TSN streams to achieve deterministic-communication. For safety-critical applications, it is crucial to ensure network operability such that during network outage or failure, the remaining network can still be operational. From this perspective, a fully distributed model can be a better choice than the other two models since it does not require a centralized controller, which may cause a single-point-of-failure. Thus, in this paper, we are interested in investigating the performance of TSN's fully distributed model for deterministic communication.

One essential task of realizing deterministic communication is the scheduling of TSN streams. Several prior works [2], [3] have addressed the scheduling problem for TSN streams in a centralized manager. However, those solutions require complex computations, as the scheduling problem in TSN is NP-hard [4], and global information about the network. Thus, it is not suitable for the fully distributed model with only locally available information and no powerful centralized server. On the other hand, IEEE standards define two mechanisms for scheduling in a distributed way: 1) Stream Reservation Protocol (SRP) is used to send a message with stream requirements to each bridge along the routing path for stream reservation [5]. 2) After receiving the message, each bridge will calculate the stream's delay bound using Urgency Based Scheduler [6], specified in IEEE 802.1Qcr Asynchronous Traffic Shaper (ATS) [7]. However, we observed that the existing solution defined by the IEEE 802.1Qcr [7] leads to a pessimistic delay bound estimation, resulting in degrading the schedulability of the TSN streams.

To conquer the above challenge, we identify and show, via a case study, that the existing stream reservation mechanism in the TSN distributed model has two problems: First, the current preemption mechanism of ATS requires TSN streams to wait for a frame transmission time at each switch when there are already lower priority streams transmitting. Those waiting times at each switch result in a pessimistic delay bound estimation. Second, ATS's delay bound estimation mechanism possesses a dependency problem. The estimation might become more pessimistic as the number of subsequent switches on the routing path increases when multiple concurrent TSN

©Copyright IEICE - APNOMS 2021

streams transmit along the same path. Then we propose a Time Division Multiple Access (TDMA)-based reservation mechanism, utilizing the concept of time-slots to solve the issue. Finally, via a series of experiments, we show that compared with the existing solution, the proposed mechanism can significantly improve the schedulability by more than 35%.

The remainder of the paper is organized as follows: Section II describes the motivation of this paper. Section III illustrates the system model and problem formulation, and then a TDMA-based stream reservation is proposed in Section IV. Experimental results are presented in Section V, and Section VI concludes the paper.

II. Motivation

In this section, we establish the motivation for improving the schedulability of the stream reservation mechanism for TSN. We first identify two problems of IEEE 802.1Qcr ATS and explain how they lead to *pessimistic delay bound* for TSN streams and result in lower schedulability. Then we present the quantitative impact of the pessimistic delay bound cased by the identified problems on the TSN stream schedulability via a series of simulations.

A. Qualitative Impact of Pessimistic Delay Bound

1) Lower Priority Streams: IEEE 802.1Qcr standard presents a formula (Equation V-9 [7] for the ATS scheme to estimate the delay bound of each stream to determine whether the arrival request of a TSN stream can join the network or not. As shown in that formula, upon a TSN request's arrival, the scheme requires calculating the interference sum of the lower priority streams along the routing path with corresponding switches. Although a TSN frame can preempt the lower priority streams, such as AVB (Audio Video Bridging) or BE (Best Effort) stream, it needs to wait for the lower priority stream to finish its current frame transmission at each switch along the routing path. In a worst-case scenario, the waiting time at each switch can be as long as the transmission time of a maximum frame, and the scheme has to add up all the waiting times on each switch when estimating the delay bound.

2) Dependency Problem: In addition, we observe that ATS's delay bound estimation mechanism exists a dependency problem. When n requests have the same routing path with corresponding switches, each subsequent switches (i.e., not ingress switch) on the routing path will count n-folds delay for each stream since they do no have accurate timing information. For example, as shown in Fig. 1, two streams simultaneously arrive at Switch 1, and they have the same routing path (Switch 1 to Switch 2) and transmission time (10μs at each switch). Then the scheme will send the two reservation requests along the same routing path to estimate these stream's delay bonds. Here, we assume that the transmission of Stream 1 will be earlier than that of Stream 2. Via utilizing the estimation formula (Equation V-9 [7] of IEEE 802.1Qcr), the ingress Switch 1 can efficiently reserve the bandwidth required to transmit the two streams, and the estimated delay bound of Stream 2 at Switch 1 is 20μs. However, the estimated delay bound of Stream 2 at Switch 2 is 40 μs, instead of 30 μs. The reason is that without the two streams' actual transmission time, Switch 2 can only assume the worst case that these two streams will arrive simultaneously. That means Stream 2 needs to wait for the transmission of Stream 1 while, in reality, Stream 2 does not need to wait since it will arrive 10 μs after Stream 1. Hence, Switch 2 will reserve double-counted bandwidth for the two TSN requests, which will lead to a *pessimistic delay bound* for each TSN stream, and things will get worse as the number of subsequent switches on the routing path increases.

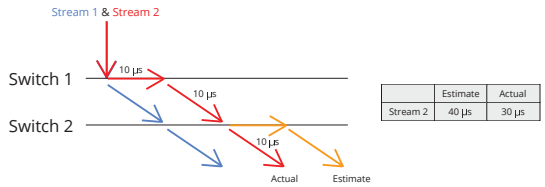

Fig. 1. Example of dependency problem

B. Quantitative impact of the pessimistic delay bound

We perform a series of simulations to quantify the impact of pessimistic delay bound caused by lower priority streams and the dependency problem on the TSN stream schedulability.

1) Simulation Setup: The network topologies are depicted in Fig. 2(a) and Fig. 2(b), which is for investigating the impacts of lower priority streams and dependency problem, respectively. The link capacity is set as 1Gbps, and there are 300 TSN steams. Each talker adopts Stream Reservation Protocol (SRP), defined [5], and Resource Allocation Protocol (RAP) [8] to check whether an arrival stream meets the latency constraints on each switch. A stream can be admitted by a switch only when its latency bound does not exceed the per-hop delay and end-to-end latency requirements. The delay bound rely on the operator's requirement. Referred to prior parameter settings [9], [10], we set the stream period, per-hop, and end-to-end delays as 150μs, respectively. The link capacity is set as 1 Gbps. The performance metric is the number of schedulable TSN streams since it allows us to have some sense of the TSN stream schedulability.

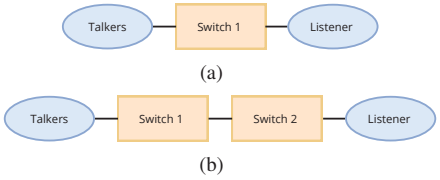

Fig. 2. (a) Topology 1 (b) Topology 2.

2) Impacts of Lower Priority Streams: First, we discuss the impacts of lower priority streams. Here we choose Topology 1 (Fig. 2(a)) as the network topology and define two scenarios: (1) **ATS**, with 300 TSN streams and one BE stream; (2) **ATS w/o BE**, with 300 TSN streams no BE stream.

Fig. 3(a) shows the impact of lower priority streams on the number of schedulable TSN streams under different packet sizes. We can see that as the packet size grows, the number of schedulable TSN streams decreases. Compared with **ATS w/o BE**, **ATS** will reduce up to about 8% schedulable TSN streams under the packet size with 64 bytes due to the impact of BE streams. The result shows that the lower priority stream (*i.e.*, BE stream) will decrease the schedulability of TSN streams.

©Copyright IEICE - APNOMS 2021

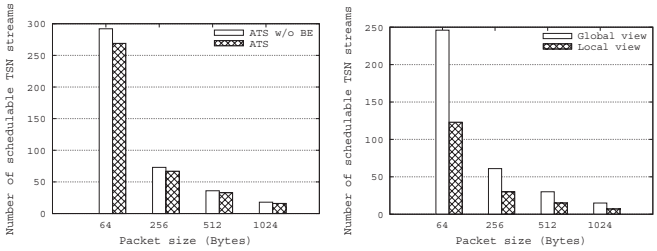

Fig. 3. (a) Impacts of lower priority streams (b) Dependency problem

3) Impacts of Dependency Problem: Then, we evaluate the impacts of the dependency problem. Here we adopt Topology 2 (Fig. 2(b)) as the network topology, where there are a set of talkers and only one listener, and each stream will transmit data from different talkers to a single listener simultaneously. In this simulation, we define two scenarios: (1) **Local view**, employs the *ATS* mechanism to calculate the delay bound for each stream request. (2) **Global view**, calculate the delay bound in global view for each switch, assuming that the exact arrival time for each stream at each switch is known. To realize the global view concept for Topology 2, we modify the Equation V-2 [7] for Switch 2 as follows:

$$d_{BU,max}(k,f) = max_{h \in F_S(k,f)} \left\{ \frac{-l_{min}(h) + l_{LP,max}(k,h)}{R(k)} + \frac{l_{min}(h)}{R(k)} \right\} \quad (1)$$

Note that the modification highly depends on the network topology. Different topologies would require specific modifications to achieve the global view. Thus, it is not a generally feasible solution.

Fig. 3(b) shows the impact of the dependency problem on the number of schedulable TSN streams under different packet sizes. We can observe that as the packet size grows, the number of schedulable TSN streams decreases. The reason is that each switch requires more bandwidth to transmit packets such that the available bandwidth of each link will significantly reduce, and thus, the number of schedulable TSN streams decreases. Scenario **Global view** can accept more TSN streams than the scenario **Local view** since the **Global view** eliminates the delay bound overestimation at Switch 2 using Equation 1. The result shows that with the current ATS mechanism (**Local view**), the schedulability can be much lower (up to 50%) than the optimal (**Global view**).

III. System Model and Problem Formulation

A. System Model

We consider a general case of the TSN network, consisting of end devices, bridges, and physical links. We model the topology as a directed routing graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$, where $\mathbb{V}$ is the set of all nodes and $\mathbb{E}$ is the set of all links. Each node in $\mathbb{V}$ is either an end device or a bridge, *i.e.*, $\mathbb{V} = \mathbb{D} \cup \mathbb{B}$, where set $\mathbb{D}$ and $\mathbb{B}$ include all end devices and bridges, respectively. Each link in $\mathbb{E}$ is a physical point-to-point full-duplex wire connection between two nodes in $\mathbb{V}$. A link cannot connect two end devices.

Each bridge has several ports with one *scheduled pool*, one BE queue, and multiple numbers of cyclic time-slots. The time-slots are for performing transmission selection and stream reservation; BE queue is for storing and transmit packets of BE streams, and the scheduled pool is for storing packets from TSN streams and transmit them according to the time-slots. Unlike the default behavior of the bridges in the TSN network, we apply the idea of dividing every egress port in time. Every egress port in the bridge has numbers of cyclic *time-slots* with the same duration τ defined by the user. The number of cyclic *time-slots* in n port of h bridge c_n^h is defined as the least common multiple of reserved streams' period. The concept of *time-slot* in bridge processing and reservation are different: (a) bridge processing, described in section IV-B, *time-slot* is for transmission selection, and (b) reservation procedure, described in section IV-A, is to check whether the stream can be reserved in that slot or not. In each port, a table called *reservedTable* stores the information about reserved streams. The content of *reservedTable* includes stream's (a) Stream ID I_i (b) packet size L_i (c) period T_i (d) offset o_i. The value o_i, which is decided during the reservation procedure, indicates the number of *time-slots* that reserved stream need to wait for when the stream arrives at the bridge.

TABLE I
Notation Table

Variable	g, h	the indexes of an end device, a bridge
	i	the index of a stream
	n	the index of a port for a bridge
	t, r	the synchronized time and link capacity
	τ	the duration of a time-slot
	c_n^h	the number of cyclic time-slots
	o_i	the offset of stream i
Set (Given)	$\mathbb{G}, \mathbb{V}, \mathbb{E}$	the sets of routing graph, vertex, and edge
	$\mathbb{D}, \mathbb{B}$	the set of all end devices and all bridges
	$\mathbb{S}$	the set of all input streams
	$\mathbb{S}_{BE}, \mathbb{S}_{TSN}$	the set of all input BE and TSN streams
Parameter (Given)	S_i, D_i	the source/destination indexes of stream i
	L_i, T_i	the packet size and period of stream i
	E_i	the end-to-end deadline of TSN stream i
	I_i	the stream ID of stream i
	A_i	the start transmission time of TSN stream i

We model the applications as a set of information that can be transmitted as TSN or BE streams. $\mathbb{S} = \mathbb{S}_{TSN} \cup \mathbb{S}_{BE}$ denotes the set of streams in the network. Associated with each BE stream $(\mathbb{S}_{BE})_i$ is the tuple of attributes: (S_i, D_i, L_i, T_i), and for each TSN stream $(\mathbb{S}_{TSN})_i$, its tuple of attributes is $(S_i, D_i, L_i, T_i, E_i, I_i, A_i)$ S_i and D_i denotes the source-node and destination-node index of the stream, respectively. Both kinds of streams are periodic with a period T_i and packet size L_i. Only the TSN stream has an end-to-end deadline E_i, stream ID I_i, and the start transmission time A_i. Table I lists the notations used throughout the paper.

B. Problem Description

We are interested in maximizing the number of accepted TSN streams when determining whether to accept or reject the request of TSN streams at runtime via tackle the problems of pessimistic delay bound in the existing ATS scheme. The determination must ensure that the end-to-end delay bounds of all accepted TSN streams do not exceed their deadlines.

The formal definition of the problem addressed in this paper is as follows:

- **Given:**

©Copyright IEICE - APNOMS 2021

– $\mathbb{G}$, the routing graph.
– $\mathbb{S}$, the set of input streams (including BE & TSN streams)

- **Decide:** Accept or reject the request of TSN streams.
- **Objective:** Maximize the number of accepted TSN streams.
- **Constraint:** The end-to-end delay bound does not exceed the deadline d_i of each TSN stream.

IV. TDMA-Based Stream Reservation

As we discover, overall TSN schedulability will decrease due to the lack of precise timing information to schedule the TSN stream for ATS. As a result, we advocate proposing a heuristic Time Division Multiple Access (TDMA)-based stream reservation, which is more suitable for the distributed TSN model and can improve overall network efficiency. To realize the TDMA-based stream reservation, we first illustrate the working process of a bridge's control plane for the stream reservation and then describe the processing of the data plane (*i.e.*, the data process on bridges). We assume all bridges support the time synchronization functionality (*e.g.*, Precision Time Protocol, PTP). This assumption is reasonable since time synchronization is essential to deterministic communication for today's TSN devices. Due to lack of space, we omit the input/output descriptions in the pseudo-codes of the following algorithms, procedures, and functions.

A. Reservation for Control Plane

The objective of the stream reservation process is to reserve the *time-slots* that can satisfy the requirement and determine the offset for each TSN request. We describe the process of stream reservation protocol as follow:

1) A talker generates a *talker advertise attribute* message, which includes the stream's information (I_i, L_i, T_i and start transmission time A_i) and TSpec information (MaxFrameSize and MaxIntervalFrames), then propagates the message along the routing path.
2) When a bridge on the routing path receive the *talker advertise attribute* message, it will (a) allocate *time-slots* for the stream, (b) record o_i, (c) add its per-hop delay bound to the TSpec attributes, (d) recalculate the start transmission time, (e) propagate message to the next hop.
3) At the end of the routing path, the listener uses the accumulated delay bound in the TSpec and the deadline of the requesting stream to determine whether to accept the stream or reject. The listener will transmit a *listener attribute* message back to the talker.

Algorithm 1 presents the pseudo-code of the reservation process. The $cycle$ of the *time-slots* needs to be adjusted according to the L_i of the requesting stream (Line 1). Then, $findTimeSlot()$ function is adopted to allocate resources for the stream, and returns an offset o_i. The offset o_i means that the stream needs to wait the offset time for data delivery when it arrives (Line 2). If the offset o_i is a non-negative integer (Line 3), which indicates that the bridge has enough resource (*time-slots*) for the requesting stream. Hence, we need to recalculate the start transmission time A_i, add per-hop delay bound to the TSpec attribute, and propagate this information to the next hop (Lines 4-6). After calculating the time-slots for the requesting stream, it will invoke the Procedure Bridge(). This procedure will inform the data plane (*i.e.*, bridge) to arrange the corresponding time-slots and offset for the stream. Then the procedure will prepare to transmit the data according to assigned time-slots calculating by (Line 7). Otherwise, if the delay bound exceeds the stream requirements, the bridge will transmit a reject message to the talker (Lines 8-9).

Algorithm 1 Stream Reservation Process

Reservation(I_i, L_i, T_i, A_i, $TSpec$, $timeSlot$, $cycle$, τ)
1: $cycle \leftarrow extendCycle(T_i, timeSlot, cycle, \tau)$
2: $o_i \leftarrow findTimeSlot(I_i, L_i, T_i, A_i, timeSlot, cycle, \tau)$
3: **if** $o_i \neq -1$ **then**
4: $\quad A_i \leftarrow (A_i + o_i \times \tau) \bmod T_i$
5: $\quad TSpec.accMaxDelay+ = o_i \times \tau$
6: $\quad$ transmit $\{ I_i, L_i, T_i, A_i, TSpec \}$ to next hop
7: **else**
8: $\quad$ reject

FUNCTION extendCycle(T_i, $timeSlot$, $cycle$, τ)
1: $newCycle \leftarrow lcm(cycle, T_i/\tau)$
2: **for** $c \leftarrow cycle$ to $newCycle$ **do**
3: $\quad timeSlot[c] \leftarrow timeSlot[c - cycle]$
4: **return** $newCycle$

The task of extendCycle() function is to adjust the $cycle$ of the *time-slots*. The $cycle$ is decided by the least common multiple of the existing streams' periods (Line 1). After extending the $cycle$, the old reserved information must be copied to the newly added $timeSlot$ (Lines 2-3). Finally, return the newest $cycle$ to the reservation process (Line 4).

FUNCTION findTimeSlot(L_i, T_i, A_i, $timeSlot$, $cycle$, τ, r)
1: $slotNeed \leftarrow L_i/\tau \times r$
2: $a \leftarrow (A_i + L_i/r)/\tau$
3: **for** $x \leftarrow 0$ to T_i/τ **do**
4: $\quad found \leftarrow TRUE$
5: $\quad$ **for** $y \leftarrow 0$ to $cycle/(T_i/\tau)$ **do**
6: $\quad\quad$ **if** not exist consecutive empty $timeSlot[y \cdot cycle + x + a]$ to $timeSlot[y \cdot cycle + x + slotNeed + a]$ **then**
7: $\quad\quad\quad found \leftarrow FALSE$
8: $\quad\quad\quad$ **break**
9: $\quad$ **if** $found$ **then**
10: $\quad\quad$ **return** x
11: **return** -1

The findTimeSlot() function is to allocate consecutive *time-slots* for the adding stream. A value $slotNeed$ is initialized as the number of slots required by the adding stream (Line 1). The arrival time (slots) a is computed by the start transmission time a_i and the packet size l_i (Line 2). Then, for each period of the stream, find the consecutive empty *time-slots* that meet its needs of bandwidth (Lines 3-8), and return a value x (a.k.a. offset o_i) (Lines 9-10). If there are not enough resources for the adding stream, return a value of -1 to represent that cannot find any *time-slot* (Line 11).

©Copyright IEICE - APNOMS 2021

Algorithm 2 The Operation of Bridge

1: $t \leftarrow 0$
2: **while** true **do**
3: **if** $inputPacket \neq \emptyset$ **then**
4: $packet \leftarrow inputPacket$
5: **if** $packet.type$ is $Reservation$ **then**
6: Reservation($packet.I_i$, $packet.L_i$, $packet.T_i$, $packet.A_i$, $packet.TSpec$, $timeSlot$, $cycle$, τ)
7: **else**
8: Allocation($packet$, t, τ)
9: Forwarding(t, τ, r)
10: $t \leftarrow t + 1$

B. Data Plane Processing

Algorithm 2 describes the process of the stream reservation request and packet processing upon a packet arrival for each bridge. The initialized time t is set as 0 (Line 1). If the input packet is a reservation type stream, it will invoke the Reservation() to allocate the time-slots for the packet (Lines 2-6). Otherwise, it means that the packet's time-slots are reserved, and the bridge tries to label the packet with corresponding time-slots and put it into a resource pool (Lines 7-8). After the packet labeling, the bridge will deliver the packet at the specific time-slots to the next hop (Line 9). Finally, current time adds with 1 (Line 10).

FUNCTION Allocation($packet$, t, τ)

1: **if** $reservedTable[I_i] \neq NULL$ **then**
2: $N \leftarrow \frac{t}{\tau}$ mod c_n^h
3: $packet.E \leftarrow (N + o_i)$ mod c_n^h
4: add $packet$ to $scheduledPool$
5: **else**
6: push $packet$ to $beQueue$

Allocation. If the packet belongs to a reserved stream (Line 1), then it will be assigned an eligibility slot E determined by current *time-slot* number, the corresponding *offset* in *reservedTable*; then it will be added to Scheduled Pool (Lines 2-4). If the packet does not belong to any reserved stream, it will be treated as a best-effort packet and be pushed into BE Queue (Lines 5-6).

FUNCTION Forwarding(t, τ, r)

1: $N \leftarrow \frac{t}{\tau}$ mod c_n^h
2: $packetList \leftarrow$ find all $packets$ in $scheduledPool$ if $packet.E = N$
3: **if** $packetList \neq NULL$ **then**
4: **for each** $packet$ in $packetList$ **do**
5: transmit $packet$
6: **else**
7: $count \leftarrow 0$
8: **while** $timeSlot[(N + count)$ mod $c_n^h] = 0$ **do**
9: $count \leftarrow count + 1$
10: $packet \leftarrow beQueue.pop()$
11: **if** $packet.length \leq count \times \tau \times r$ **then**
12: transmit $packet$

Forwarding mechanism. According to the current *time-slots* duration (Line 1), it will pick a corresponding packet from Scheduled Pool to transmit data (Lines 3-5). If there is no eligible packet (Line 6), it will calculate the remaining bandwidth for the best-effort packets (Lines 7-9). Finally, the bridge will select a packet from the BE Queue to transmit data (Lines 10-12).

V. Experimental Results

In this section, we have performed two sets of experiments to evaluate the performance and overhead of the proposed TDMA-based stream reservation approach.

A. Experimental Settings

We consider two different network topologies for each experiment: (1) a simplified ring topology of TSN-based industrial factories, removing a switch from left and right sides of ring topology respectively from Figure 7 in [11], and (2) a mesh-like topology of Orion Multi-Purpose Crew Vehicle [12]. We define two types of distribution of test cases: (1) Normal distribution and (2) Uniform distribution to simulate real-world stream traffic. The parameter *start transmission time* in both test cases follows Poisson distribution for more realistic simulations. The TSN stream period ranges from 100 to 1000μs [10]. The deadline for each stream is equal to its period. The data size of each stream ranges from 30 to 100 Bytes. The TDMA-based time-slot duration τ is set as 5μs. The baseline for the comparison is the Asynchronous Traffic Shaping (ATS) [7, Eq.V-2], which uses Resource Allocation Protocol (RAP) to provide dynamic reservations for streams. Each stream's per-hop deadline is set as $\frac{T_i}{hop\ count}$ for the ATS scheme. For the performance evaluation, the number of accepted streams (schedulability) is the metric. Then we choose the time required by the schedule computation (computation time) as the metric for the overhead evaluation. Each report's result is an average of 50 independent runs on a desktop computer with AMD Ryzen 7 1700 CPU and 16GB RAM.

B. Schedulability Evaluation

Here we are interested in evaluating the performance of the proposed approach under the ring [11] and orion [12] topology. In this experiment, 3000 TSN streams are deployed with random properties: talker node, listener node, TSN data size, and period. These streams attempt to reserve network resources successively. For each reservation process: (1) for ATS, each of the bridges along the routing path compares the per-hop delay bound d_{max} with its per-hop deadline, and (2) for TDMA-based, the listener compares the accumulated delay bound with stream's deadline. If each bridge and listener accepts the adding stream, the network will also accept it.

As shown in Fig. 4, we can see that the schedulability of TDMA-based outperforms the ATS. The reason is that streams are reserved in dedicated time slots; that is, the transmission of streams is always at a fixed time. By using the TDMA-based approach, bridges can compute the delay bound more accurately and solve *lower priority streams* and *dependency* problems. In the ring topology (Fig. 4), the ATS approach can only reserve around 1200 TSN streams while the TDMA-based approach can deploy more than 2500 TSN streams to the network. The ATS approach's schedulability in the topology

©Copyright IEICE - APNOMS 2021

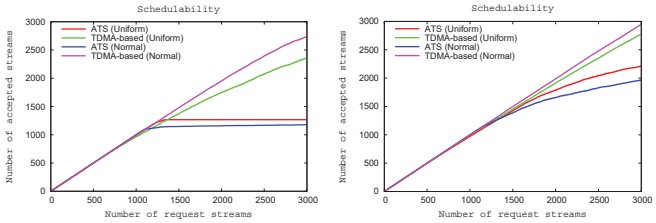

Fig. 4. (a) Schedulability of ring topology (b) Schedulability of Orion CEV topology

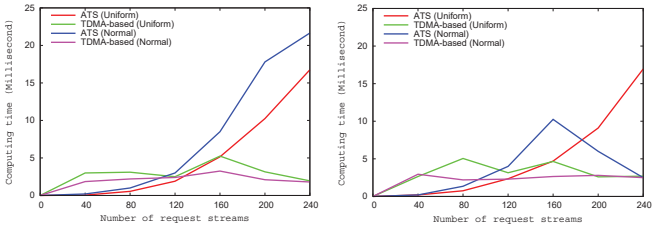

Fig. 5. (a) Computation time of ring topology (b) Computation time of Orion CEV topology

(Orion) is better than a small topology. It is because there are more bridges and paths that can hold streams. However, the TDMA-based approach still outperforms the ATS when deploying more than 1500 TSN streams to the network. The distribution of streams' properties causes different results in two approaches. For the TDMA-based approach, because of using a greedy strategy and not considering future streams' scalability, the Normal distribution test case, which has less urgent streams (lower deadline), has better schedulability. On the other hand, the ATS approach under Normal distribution has worse schedulability because more medium data sized streams occupy the bandwidth. The results verify the performance of the proposed approach as it can achieve similar or better schedulability than ATS. Moreover, the gap becomes broader as the number of stream requests increases.

C. Computation Time

Here we are interested in examining the computation overhead of the scheduling approaches. We compare the computation times of our approach with ATS under different numbers of request streams in the bridge. As shown in Fig. 5, we can observe that, as the number of request streams increases, ATS requires more computation time. It is because, at each reservation, each of the bridges compares not only the adding stream's delay bound but also every already deployed stream's to make sure they will not exceed the per-hop deadline guarantee. Thus, the more streams deployed in the bridge, the more time need to check and compute. Besides, the result shows that, in some cases, the bridge does not need to spend much computation time even if the deployed streams increases. The reseason is that the bridge's bandwidth is full so that the bridge only checks a few streams before rejecting the request. The computation time of the TDMA-based approach is more stable than ATS since it only checks the capacity of time-slots and does not need to check the existing deployed streams. The results showcase the superiority of the proposed approach on the computation overhead compared with ATS.

VI. Conclusions

In this paper, we focus on the scenario of the fully distributed model of TSN networks and figure out, via a case study, that the existing solution has two problems and will result in a pessimistic delay bound. We then present a TDMA-based stream reservation mechanism to attain a precise delay bound for each stream to solve the problems. Finally, experimental results show that compared with the current solution (ATS), the proposed algorithm can accommodate more TSN streams and also provide a low computation time at bridges.

Acknowledgement

Chuang's work was supported by National Pingtung University under Grant C11026 and by MOXA Networking Co., Ltd. Shih's work was supported in part by the Ministry of Science and Technology under Grants 109-2222-E-194-003-MY3, by AIM-HI of National Chung Cheng University, by MOXA, and by Qualcomm. Pang's work was supported in part by Ministry of Science and Technology under Grants 108-2221-E-002-069-MY3 and 110-2221-E-002-071-MY3, by National Taiwan University under Grant 110L892702, by Ministry of Economic Affairs under Grant 109-EC-17-A-02-S5-007, by Qualcomm Technologies Inc., and by MOXA Networking Co., Ltd.

References

[1] "Ieee standard for local and metropolitan area networks–bridges and bridged networks – amendment 31: Stream reservation protocol (srp) enhancements and performance improvements," *IEEE Std 802.1Qcc-2018 (Amendment to IEEE Std 802.1Q-2018 as amended by IEEE Std 802.1Qcp-2018)*, pp. 1–208, Oct 2018.

[2] 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*, 2016, pp. 203–212.

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

[4] 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.

[5] "Ieee standard for local and metropolitan area networks–virtual bridged local area networks amendment 14: Stream reservation protocol (srp)," *IEEE Std 802.1Qat-2010 (Revision of IEEE Std 802.1Q-2005)*, pp. 1–119, 2010.

[6] J. Specht and S. Samii, "Urgency-based scheduler for time-sensitive switched ethernet networks," in *2016 28th Euromicro Conference on Real-Time Systems (ECRTS)*, 2016, pp. 75–85.

[7] "Ieee draft standard for local and metropolitan area networks–bridges and bridged networks amendment: Asynchronous traffic shaping," *IEEE P802.1Qcr/D2.3 (Draft amendment to IEEE Std 802.1Q-2018), May 2020*, pp. 1–152, May 2020.

[8] "Ieee standard for local and metropolitan area networks - bridges and bridged networks - amendment: Resource allocation protocol (rap)," *IEEE, P802.1Qdd*, 2019.

[9] C. C. Chuang, T. H. Yu, C. W. Lin, A. C. Pang, and T. J. Hsieh, "Online stream-aware routing for tsn-based industrial control systems," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2020, pp. 254–261.

[10] Time-sensitive networks for flexible manufacturing testbed-description of converged traffic types. [Online]. Available: https://www.iiconsortium.org/pdf/IIC_TSN_Testbed_Traffic_Whitepaper_20180418.pdf

[11] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. Elbakoury, "Performance comparison of ieee 802.1 tsn time aware shaper (tas) and asynchronous traffic shaper (ats)," *IEEE Access*, vol. 7, pp. 44 165–44 181, 2019.

[12] Z. Zheng, F. He, H. Li, and J. Lu, "Design optimization of time-triggered ethernet based on routing and scheduling strategy," in *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, 2019, pp. 1–7.

©Copyright IEICE - APNOMS 2021