A Formal Method for Evaluating the Performance of TSN Traffic Shapers using UPPAAL

Wang Guo[†], Yanhong Huang[*†], Jianqi Shi[†], Zhe Hou[‡], Yang Yang[†]
[†]National Trusted Embedded Software Engineering Technology Research Center,
East China Normal University, Shanghai, China
{wang.guo,yang.yang}@ntesec.ecnu.edu.cn
{yhhuang, jqshi}@sei.ecnu.edu.cn
[‡]Griffith University, Australia
z.hou@griffith.edu.au

***Abstract*—Deterministic low latency network plays an important role in control systems, such as industrial control system(ICS), Advanced Driver Assistance Systems (ADAS), et al. There are quite tight timing requirements in these areas. The IEEE 802.1 Time-Sensitive Networking (TSN) task group has proposed several traffic shapers to satisfy real-time communications requirements. Traditionally, the performance of TSN is analyzed by simulations, whereas these methods cannot cover all corner cases. With limited knowledge, this paper firstly presented formal models of the TSN's time-aware and peristaltic shapers using UPPAAL, tactically solving the problem mentioned above. Afterwards, we verified some real-time critical properties of the shapers models, of which the results could evaluate whether these shapers are able to satisfy strict timing requirements or not. Based on the shapers models, we can make a further discussion about the performance of time-critical frames combining the preemption mechanism. Preemption is used to handle the conflict in scheduling data frames with different priority levels. Moreover, we can also analyze the resource utilization and transmission latency. Under the time properties analysis and verification of TSN traffic shapers, we can provide engineers with an accessible reference which may assist them in developing the TSN.**

***Index Terms*—IEEE 802.1TSN, Preemption, Performance Analysis, Real-Time, UPPAAL**

I. Introduction

With the rapid development of various areas including intelligent manufacturing, autonomous vehicles, the Internet of things, and other automation fields, bandwidth and deterministic delay requirements are becoming highly necessary for the network. Unfortunately, traditional network like CAN or FlexRay have been unable to meet the requirements. Instead, Ethernet has gradually become a backbone network in those areas to satisfy the scalability and ultra-high-speed requirements. It also supports complex topology [1]. However, The deterministic timing guarantees for critical traffic are essential to the complex temporal behavior of Ethernet. Therefore, 8 traffic classes have been defined in the standard Ethernet IEEE 802.1Q [2]. These classes can be scheduled in order by a strict priority in the non-preemptive (SPNP) scheduler at each output port of each switches and endpoints. But this order introduces a major source of delayor the reason that critical traffic might be blocked by lower priority traffic.To cope with this problem, the IEEE 802.1 TSN Task Group develops several shapers , such as time-aware (TAS) and peristaltic shapers (PS) [3]. These shapers are capable of accommodating hard real-time traffic with deterministic end-to-end delays. The TSN network stations communicate up to 18 times faster than with any protocol available on the market today. This opens up new possibilities in areas such as tightly synchronized motion and control applications.

Although a number of common methods have appeared for evaluating network performance, there is a lack of effective methods for real-time performance analysis. Among those methods for evaluating network performance, the most common one is simulation [4], which is used to assess a network's response to a given set of stimuli. However, due to the extremely high reliability requirements, simulations often require long simulation runs and usually do not expose all corner cases. Another common method is real-time calculus. It's impossible to automate the verification, and only the worst-case scenario can be assessed, which can lead to rather pessimistic conclusions for engineers.

To address these challenges, we recommend applying *model checking* [5] in this paper. *Model checking* is one of the most important and successful automatic techniques for verifying highly sensitive systems such as security protocol. It exhaustively searches the state space of the system to cover all corner cases. In order to analyze the timing requirements about TSN, we specifically adopt a timed model checker UPPAAL [6] to model the TSN traffic shapers.

UPPAAL is an integrated tool for modeling, simulation and verification of real-time systems [7]. It is well suited for verifying such systems that can be modeled as nondeterministic processes [8]–[10]. During the modeling of systems, the simulator may detect possible problems existed. The verifier may check the safety and liveness properties of the system by searching the system state space on-the-fly.

With UPPAAL, we model the shaping process at each router node. Each process is modeled as a timed automaton. In particular, *Window automata* deal with the time slots division of the cycle. *Traffic automata* deal with the data forwarding processes. Moreover, we provide the models of the traffic shapers combining preemption mechanism. We validate our

*Corresponding author.

models using the simulator and verify some properties of our models. Furthermore, we conduct a formal analysis that focuses on resource utilization of networks and end-to-end delay of time-critical frames under four scenarios respectively, i.e., TAS, TAS with preemption, PS, PS with preemption.

The main contributions of this article can be summarized as follows.

1) **Modeling and Verification:** We build formal models of the TSN time-aware and peristaltic shapers mechanisms as well as those with preemption mechanism using UPPAAL. Based on the shapers models, we verify several properties, i.e., deadlock-free, reachability of preemption, and low latency.
2) **Timing Analysis:** We analyze the resource utilization of shapers and the transmission latency of real-time critical frames. We provide the formal performance analysis of TSN traffic shapers and preemption mechanism.

Related Work There have been some work on the analysis of TSN in related industries and academia. Bello et al. [11] gave an overview of TSN in industrial systems. This paper identified validation and verification as one of the TSN research challenges. Luxi Zhao et al. [12] proposed a network calculus-based approach to determine the worst-case end-to-end delays of audio/video bridging traffic with non-preemption and preemption. Peng Zhang et al. [13] presented an analysis framework of TSN to analyse the worst-case latency through real-time calculus. Although the above two approaches can calculate the worst-case delay by arithmetic, they rely too much on parameters and cannot automatically calculate. [14] used network simulation tool NeSTiNg to investigate the TSN scheduling mechanism. However, it's very time consuming and can not cover all cases. In [15], IEEE 802.1i security protocol is modeled and verified based on timed automata using model checker UPPAAL. It could check correctness of the communication protocol automatically and provide better security. In this article, we recommend applying model checking.

In our paper, we consider the real-time performance of time critical frames under the time-aware and peristaltic shapers with preemption and non-preemption. To the best of our knowledge, there is no other study to model and verify the TSN's TAS and PS for evaluation.

Paper Organization The remainder of this paper is organized as follows: a brief overview of TSN protocol and UPPAAL is given in Section II. In Section III, we give the formal models of the TSN traffic shapers using UPPAAL. Section IV presents experimental configuration, verification and formal analysis of the TSN shapers. Section V concludes this article.

II. Preliminaries

The IEEE 802.1 TSN has defined new mechanisms that enable real-time systems using Ethernet technologies. Specifically, TSN defined time-aware and peristaltic shapers to allow the convergence of low-latency scheduled traffic (ST) and standard best-effort (BE) Ethernet traffic on the same network.

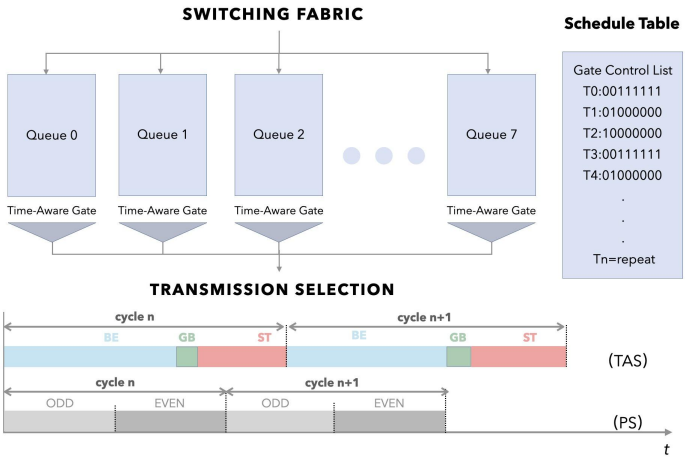

Fig. 1. Operation of Shapers.

Data frames for scheduled traffic are real-time frames, smaller in size, but more demanding in terms of real time. Data frames for best-effort traffic are larger in size, but less demanding for real-time.

As shown in Fig.1, there are eight queues to accommodate the two types of data frames mentioned above. A time-aware gate on each queue port controls the timing of data frame forwarding. The number 0 in the scheduling table on the right indicates that the gate is closed and transmission is suspended. The number 1 indicates that the gate is open and transmission is allowed. In the module of transmission selection algorithm, the engineering developer can choose various scheduling algorithms depending on the latency needs of different applications.

The bottom part of Fig.1 shows the time slots division of the periodic time cycle shaped by TAS and PS. The shaping rules in the corresponding time slots are described as follows.

A. IEEE 802.1 Qbv: Time Aware Shaper

Time-aware shaper defines a mechanism for time-driven control and scheduling of data frames. It is designed to separate the communication into fixed length and repeating time cycles. In this paper, each cycle is further divided into three time slots. As shown in the *TAS* at the bottom of Fig.1, the time slot *BE* allows for standard best-effort Ethernet traffic to be sent and the time slot *ST* only allows the transmission of low-latency scheduled traffic with high priority. The time slot *GB* is called guard band which is explained in the following scenario.

As shown in Fig.2, the top row shows a typical conflict scenario. A new best-effort (**BE**) frame transmission is started in time slot *T0* (BE time slot) and it infringed the following time slot *T1* (ST time slot). By partially or completely blocking the time-critical ST time slot, the real-time frames **ST1** and **ST2** can be delayed up to the point where they cannot meet the application requirements any longer. In the non-preemptive mode (NP in the second row), time-aware shaper handles this conflict by setting a guard band in front of *T1*. The duration

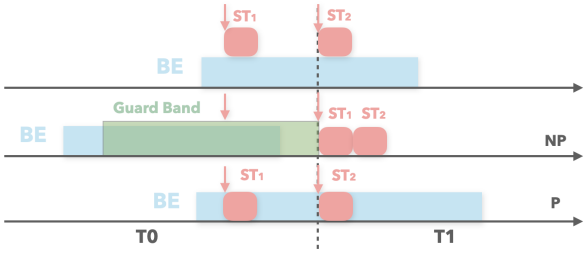

Fig. 2. Time-Aware Shaper (TAS)

of the guard band is as long as the length of the maximum best-effort frame. None of new frame transmission is allowed to start within guard band. While the guard bands manage to protect the time slot with high-priority traffic, they also have a significant drawback. They waste bandwidth for traffic on the Ethernet link.

B. IEEE 802.1 Qch: Peristaltic Shaper

Peristaltic shaper provides a traffic shaping method that can deliver deterministic and easily calculated latency for time-sensitive traffic. As shown in the *PS* at the bottom of Fig.1, peristaltic shaper also divides the transmission time of data frames into fixed length and consecutive time cycles. The peristaltic phase is dependent on the required maximum end-to-end latency the width of real-time frames. *PS* divides a time cycle into two equal-length slots (*ODD* and *EVEN* time slots). In addition, each time slot allows for both scheduled traffic and best-effort traffic to be sent. *PS* specifies that data frames arriving in *ODD* time slot are required to wait until the following *EVEN* time slot to send, and vice versa [16].

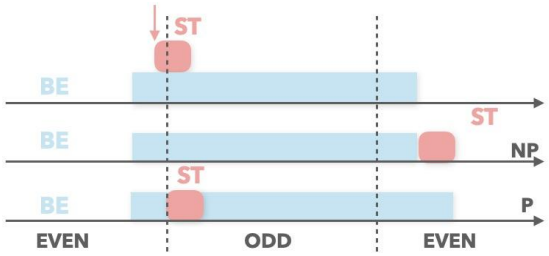

Fig. 3. Peristaltic Shaper (PS)

As shown in Fig.3, the top row shows a typical conflict scenario. When a real-time frame **ST** arrives before the start of the *ODD*, there is a best-effort frame **BE** being transmitting. Unfortunately, the frame **BE** is too large to fit into the time slot *ODD*. The frame **ST** will be blocked, because the transmission of the frame **BE** cannot be interrupted. In non-preemptive mode (NP in the second row), the frame **ST** is forwarded until the frame **BE** to complete transmission. The worst-case **ST** frame delay can be two cycles.

C. IEEE 802.1 Qbu: Frame Preemption

IEEE 802.1Qbu has specified the frame preemption technology [17]. Preemption allows one or more express (higher-priority) frames to interrupt the transmission of a preemptable (lower priority) frame. It also defines procedures to resume the transmission of the preemptable frames once the express frames have been transmitted.

To mitigate the negative effect from the guard bands in Qbv and improve the real-time performance of frame **ST** in Qch, we recommend the frame preemption technology. The bottom row (P) in Fig.2 indicates the preemption behavior of the real-time frames **ST1** and **ST2** in Qbv. They can be forwarded as soon as they arrive. Likewise, the bottom row (P) in Fig.3 means that the frame **ST** is allowed to be forwarded at the start of the following time slot *ODD* with preemption in Qch. As a result, preemption can reduce the end-to-end latency of real-time frames.

D. UPPAAL Model Checker

The core functionality of UPPAAL is model-checking of hard real-time properties on timed automata models. It is supported by a verifier and simulator, which enables manual and random tracing through the model. All the clocks progress synchronously. The simulator can be used in three ways: 1) the user can run the system manually and choose which transitions to take, 2) the random mode can be toggled to let the system run on its own, or 3) the user can go through a trace to see how certain states are reachable. By the ways, we can cover all corner cases. The main purpose of a model checker is to verify the model against requirement specifications.

UPPAAL gives a pseudo-formal semantics for the requirement specification language. UPPAAL uses $A[]$ to represent AG of CTL. $A[]\, p$ evaluates to true only when every reachable state satisfies predicate p. The property $A <> p$ to represent AF of CTL, which evaluates to true only when all possible transition sequences eventually reach a state satisfying p. the property $E <> p$ to denote EF of CTL that evaluates to true if and only if there is a sequence where there exists a reachable state satisfying p. The state property *deadlock* evaluates to true for a state only when there is no action successor.

III. Modelling the Traffic Shapers

In this section, we formally model the processes of the TSN time-aware and peristaltic shapers using UPPAAL.

A. Abstracting Data Structures in TSN Protocol

In the text below, We define abstract data structures in the TSN protocol. The terminal station generates data frames that are sent to the switch.

Definition 1 (Data Frame)**.** A data frame F is defined as a tuple $(FrameSize, ReceiveTime, Class, FrameInterval)$, where each component is explained below.

- *FrameSize* indicates the byte size of the data frame.
- *ReceiveTime* is the arrival time of the frame.
- *Class* gives the priority of the frame, i.e., ST and BE frames.
- *FrameInterval* indicates how often the data is sent.

Definition 2 (Queue)**.** A queue Q is defined as a quadruple (*Array[n], Class, Head, Tail*), where

- *Array[n]* indicates a sequence of frames in the queue, and the length of the sequence is *n*.
- *Class* is the priority of the queue, ranging from 0 to 7.
- *Head* is the index of the first element in the queue.
- *Tail* is the index of the last element in the queue.

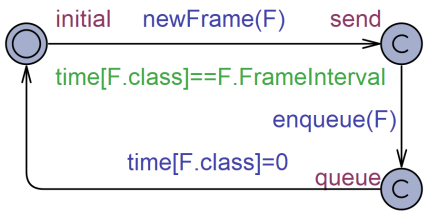

Fig. 4. Terminal Station

As shown in the Fig.4, we create an automaton that simulates the process of data producing. In the *initial* location, if the global clock variable *time[*]* equals to the value of *F.FrameInterval*, the terminal station starts to generate a frame. *newFrame(F)* is a function used to generate a frame. Then the frame is queuing in the switch. *enqueue(F)* is responsible for queuing the frame. The process will repeat according to the frame interval.

B. Transmission Substandards Models

We abstract the shaping behavior of the time-aware and peristaltic shapers into two parts. One part is about the time slots division, which is represented as *Time Window Automata*. The other part is about controlling the data traffic transmission within the corresponding time slots. The part is described as *Traffic Automata*. A complete shaping process is described as a network of timed automata with the two parts above.

TABLE I
ABSRTACT TIMING STATES IN SHAPERS

States	Description
BE	Best-effort traffic transmission time slots
GB	Guard band time slots
ST	Scheduled traffic transmission time slots
ODD/EVEN	Odd/Even time slots
preempted	Preempting transmission time slots
resumeBE	Resuming transmission time slots
waitTimeout	Waiting for timeout error

Locations: We extract some states of time slots in the communication. Table I lists the time slots states {*BE, GB, ST*} under TAS rules, states {*ODD, EVEN*} under PS rules and states {*preempted,resumeBE*} under the preemption behavior. The state *BE (ST)* allow for only *BE(ST)* frames transmission. The state GB represents time slots Guard Band. There is no new frame allowed to start within *GB*. The states *ODD (EVEN)* means that the current time slot is marked as ODD (EVEN). The mark is used as a basis for determining whether data frames can be forwarded or not. The state *preempted* indicates the point where the preemption happen, and state *resumeBE* is for resuming BE frames transmission. The state *waitTimeout* warns that the the waiting time of real-time frames has expired.

Channels: *Time Window Automata* and *Traffic Automata* synchronise over shared channels and exchange data via shared variables. Table II lists several channels. The channels act as synchronization signals to trigger the transmission of related frames. Further explanation will be given later.

TABLE II
ABSTRACT CHANNELS IN TSN PROTOCOL

Channels	Functionalities
tranBE	Sync signal used to transmit best-effort frames
tranST	Sync signal used to transmit real-time frames
preempt	Sync signal used to trigger preemption
resume	Sync signal used to trigger resumption transmission

*1) **Modeling the TAS mechanism**:* This section presents the modeling process of TSN's Time-aware shaper (TAS), including *TAS Window Automaton* and *TAS Traffic Automata*.

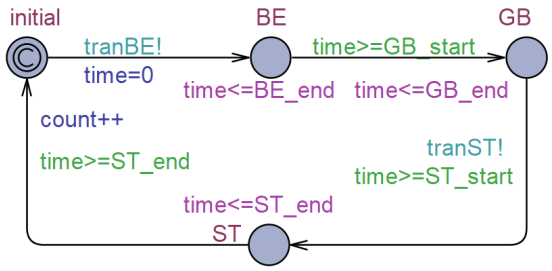

Fig. 5. TAS Window automaton

TAS Window Automaton: The time slots division rules for TAS are given in part A of the preliminaries before. As shown in Fig.5, the TAS window imitates a time cycle throughout the system. We denote the three time slots in the cycle with locations *BE*, *GB*, and *ST*. The variables *GB_start* and *ST_start* are the start moments of time slots *GB* and *ST*, respectively. The variables *BE_end, GB_end* and *ST_end* are the end moments of time slots *BE, GB,* and *ST*, respectively.

The TAS window starts in *Initial* location, where the global clock variable *time* records the elapsed time. The window determines whether to move by comparing the current *time* with the starting values of different time slots. When the window enters the BE (ST) location, the *BE (ST) Traffic Automaton* gets the permission *tranBE!(tranST!)* to transmit with *tranBE?(tranST?)* synchronisation. The variable *count* records the total number of time cycle and can be used to calculate the resource utilization.

TAS Traffic Automata: The rules for the data traffic transmission have been given in part A of of the preliminaries above. Fig.6 displays the *ST Traffic Automaton* in TAS.

Firstly, the **ST** frames are in the *wait* location, where they wait to receive the permission (*tranST?*). By the waiting delay *waitDelay*, we can make a judgment that whether the critical

frames meet the real-time performance requirements or not. In the *preST* location, if the waiting delay is greater than the maximum, then the error warning *waitTimeout* will be issued. If the waiting delay is acceptable, then function *Empty()* will check that whether the queue is empty or not. If it is not empty, then the frames is started to transmit.

Secondly, we define the variable *tranDelayST*, which indicates the length of time to transmit a single **ST** frame. The clock variable *watch* records the duration of the frame transmission. By comparing *watch* and *tranDelayST* in *sendST* location, we can see whether a single **ST** frame is completely transmitted or not. The variable *totalTimeST* records the total transmission duration of ST frames in the whole task. It is used to analyze resource utilization later.

Finally, in the *tranSuccess* location, the ST traffic automaton determines whether to move to the next time slot by comparing *time* with *ST_end*. The transmission process of the *BE Taffic Automaton* is similar to this, so we do not repeat it.

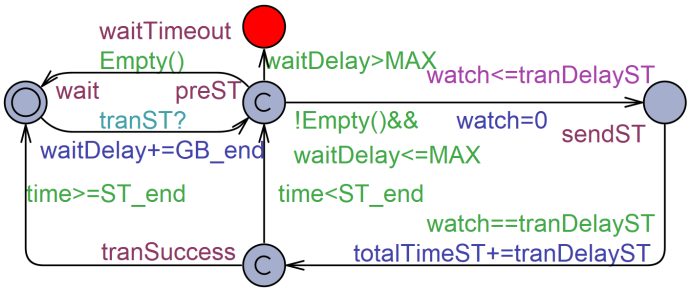

Fig. 6. ST Traffic automaton in TAS.

*2) **Modeling the PS Mechanism**:* This section presents the modeling process of TSN's peristaltic shaper(PS).

PS Window Automaton: The time slots division rules for PS are given in part B of the preliminaries before. As shown in Fig.7, the PS window imitates a time cycle throughout the system. There are two equal length time slots in every cycle, namely *ODD* and *EVEN*. *count* records the slot tag. *SLOT* (*CYCLE*) is the length of one time slot (cycle). By comparing the elapsed *time* with the *SLOT* and *CYCLE*, the window enables the rotation of time slots. Once the window switches the time slot, the signal *judege1!* synchronises with *judge1?* in the traffic automata. Then the traffic autoamta will check if frames meet the transmission conditions or not.

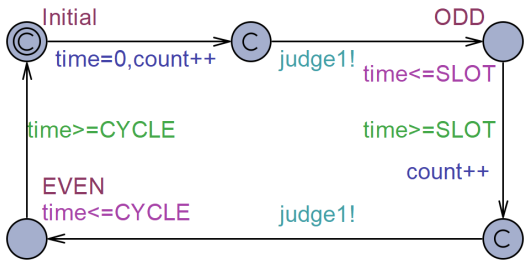

Fig. 7. PS Window automaton.

PS Traffic Automata: The transmission rules have been given in part B of the preliminaries above. Fig.8 displays the *ST Traffic automaton* in PS.

Firstly, the **ST** frames are in the *wait* location. **Step 1**, the function *Empty()* detects if the queue is empty or not. If it is empty, waiting again. If it is not empty, frames are ready in *preST* location. The method to evaluate the wait timeout is similar to the one above in TAS window. **Step 2**, when the time slot is switched (*judge1?*), or the frame being transmitted is finished (*judge2?)*, frames in the queue will check themselves to see if they meet the sending conditions ($count! = ReceiveTime$) or not. At this point, if the conditions are met, then frames go to the next step. **Step 3**, variable *lock* indicates whether the network channel is occupied or not. When the *lock* equals *0*, the channel is idle.

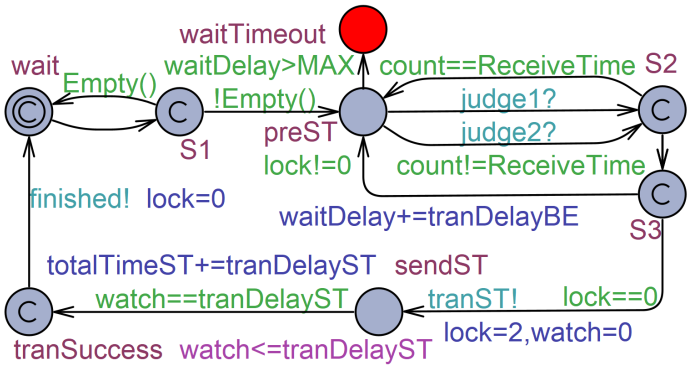

Fig. 8. ST Traffic automaton in PS.

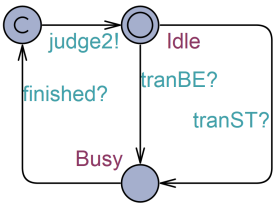

Fig. 9. Gate in PS.

Secondly, frames transmission starts in *sendST* location. **ST** frames occupy the network channel by *lock=2*. The signal *tranST!* synchronise with *tranST?* in *gate*, shown in Fig.9. The role of *watch* and *totalTimeST* here are the same as the ones of the *ST Traffic Automaton* in TAS above.

Finally, in *tranSuccess* location, the frames transmission are completed and unlock the network channel. The signal *finished!* informs the *gate* to go to *idle*. The transmission process of the *BE Traffic Automaton* is similar to this, so we do not repeat it.

*3) **Modeling the TAS with Preemption Mechanism**:* This section presents the modeling process of time-aware shaper combined with the preemption mechanism in TSN. We imitate the preemption behavior based on the TAS automata and create the automata for the preemption process.

TAS with Preemption Window Automaton: When TAS is combined with preemption, the guard band mechanism is

no longer used here. The time cycle is divided into *BE* and *ST* slots. Preemption may occur in the scenario. When the **ST** frame arrives during the *BE* slot, there is a **BE** frame being transmitted.

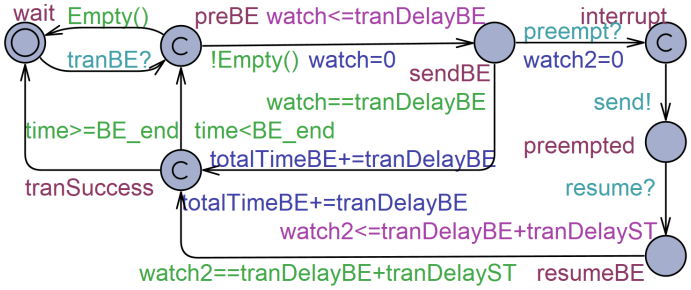

Fig. 10. BE Traffic automaton in TAS with Preemption.

TAS with Preemption Traffic Automata: The preemption rules have been given in part C of the preliminaries above. Fig.10 demonstrates that how frame preemption works based on *BE Traffic Automaton* in TAS. The transmission process of **BE** frames is similar to the transmission process of **ST** frames in TAS described above. During the process of sending a **BE** frame, if the **ST** frame arrives, it synchronises with the *BE Traffic Automaton*. This is done by the channel synchronisation *preempt?* on the transition to *interrupt* location, where the **BE** frame is suspended.

Then the **ST** frame starts transmission and the **BE** frame enters the preempted state. When the **ST** frame transmission is completed, the interrupted frame transmission is resumed in *resumeBE* location. The following process is similar to the *BE Traffic Automaton* in TAS described above. The rest of automata are not shown here due to the space issues.

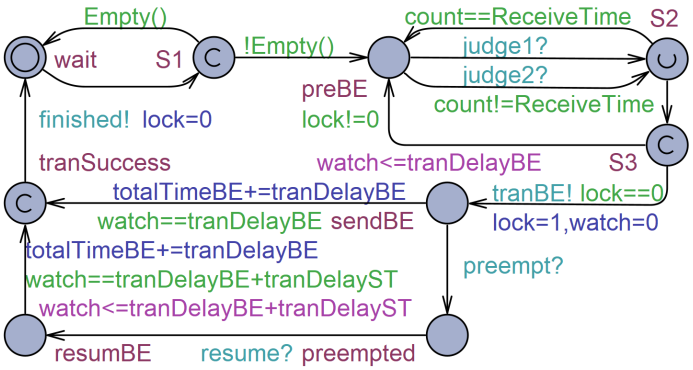

Fig. 11. BE Traffic automaton in PS with Preemption.

*4) **Modeling the PS with Preemption Mechanism**:* This section presents the modeling process of peristaltic shaper combined with the preemption mechanism in TSN.

For PS with preemption window, the time slot rules remain unchanged.

PS with Preemption Traffic Automata: The transmission process of **BE** frames is similar to the transmission process of **ST** frames in PS described above. Fig.11 demonstrates that how frame preemption works based on *BE Traffic Automaton* in PS. The preemption process is similar to that in TAS with preemption above.

TABLE III
TRAFFIC CLASSES.

Class	ST	BE
FrameSize	128B	256B
TransTime	$12\mu s$	$25\mu s$
FrameInterval	$200\mu s$	$125\mu s$
Priority	HP	LP

IV. VERIFICATION OF SHAPERS MODELS

This section validates and verifies our traffic shapers models by using the simulator and verifier in UPPAAL, respectively.

A. Experimental Configuration

In our experimental study, we consider two classes of data frames: ST and BE frames. Table III shows the parameters of these frames, whose explanations can be found as follows:

- *FrameSize* indicates the frame size in bytes. The size is made up of the payload of a message plus its header, i.e., $Size = Payload + Header$.
- *TranTime* denotes the frame transmission time (or the length), which is calculated by the size of the frame and the network bandwidth. In this paper, we assume the bandwidth is 100 Mbps.
- *FrameInterval* indicates how often the data is sent.
- *Priority* HP stands for high priority and LP stands for low priority.

According to IEEE 802.1 TSN, the period of cycle in TAS is 500 μs. The length of guard band equals the maximum-sized best-effort frame, $Lgb = 25\ \mu s$. IEEE 802.1TSN specifies the maximum end-to-end latency for real-time frames over 5 hops to be $100\mu s$.

B. Verifying Fundamental Properties

To vlidate that the model in this paper follows the rules of shapers, we define properties 1-7. To verify that the real-time frames meet the real-time performance requirements, we define property 8. The verification results and analysis are also presented in Table IV.

Property 1 (Deadlock-freedom (DF))**.** *This property ensures that the system never moves to a (deadlock) state where no moves are possible. A deadlock may lead to data loss or even system breakdown. It can be specified in CTL as*

$$A[\ ]\ not\ deadlock$$

This formula states that, for all paths, the system will never enter a deadlock state.

Property 2 (Frame Mutual Exclusion (FME))**.** *This property ensures that the shaping mechanism does not transmit frames*

of different priorities at the same time. We specify this property as the conjunction of the below formulae:

$$TAS: A[\,]\ not\ (window.BE\ and\ window.ST)$$
$$PS: A[\,]\ not\ (window.ODD\ and\ window.EVEN)$$
$$PS/TAS: A[\,]\ not\ (ST\_traffic.sendST\ and\ BE\_traffic.sendBE)$$

The first formula indicates that the time slot *BE* state and *ST* state of process *window* must not occur at the same time in TAS. The second formula means that the time slot *ODD* state and *EVEN* state of process *window* must not occur at the same time in PS. The third formula indicates that the best-effort and scheduled traffic must not occupy the communication channel at the same time. *BE_traffic* and *ST_traffic* denote the traffic automata.

Property 3 (Liveness of Forwarding Processes (LFP))**.** *It is crucial that the forwarding process of a data frame is eventually completed; otherwise, there might be data loss. We specify this property as the conjunction of the formulae below.*

$$PS/TAS: A <>\ (ST\_traffic.sendST\ imply\ ST\_traffic.tranSuccess)$$

With the above formula, the model checker can verify if the frame has finished the forwarding process or not.

Property 4 (Non-interleaving (NI))**.** *This property guarantees that the BE data frames cannot be transmitted within the ST interval, and vice versa. We formulate this property as below.*

$$TAS: A[\,]\ window.ST\ imply\ not\ BE\_traffic.sendBE$$

The formula states that **ST** frames cannot be transmitted during time slot *BE*.

Property 5 (Sequentiality of Traffic (ST))**.** *This property checks that when a data frame has arrived, it must wait until a later time slot to be forwarded. We express this property in CTL below.*

$$PS: A <>\ ST\_traffic.sendST\ imply\ (ReceiveTime\ !=\ count)$$

Property 6 (Reachability of Preemption (RP))**.** *When the transmission of low-priority frames blocks the transmission of real-time critical frames, the latter may interrupt the former and take precedence over the transmission. We formulate this event as below.*

$$TAS/PS: E <>\ BE\_traffic.preempted\ and\ ST\_traffic.preemption$$

The above formula indicates that preemption events can occur during the BE frames transmission.

Property 7 (Low Latency (LL))**.** *IEEE 802.1TSN specifies the maximum end-to-end latency for time critical frames over 5 hops as 100μs. We formulate the latency requirement as below.*

$$TAS/PS: A[\,]\ not\ (ST\_traffic.waitTimeout)$$

This property checks whether the ST frames meet the latency requirement. We can use this property to verify concrete real-time performance.

C. Verification Results

TABLE IV
VERIFICATION RESULTS

Category	TAS	PS	TAS+Qbu	PS+Qbu
DF	YES	YES	YES	YES
FME	YES	YES	YES	YES
LFP	YES	YES	YES	YES
NI	YES	/	NO	/
ST	/	YES	/	YES
RP	NO	NO	YES	YES
LL	NO	NO	YES	YES

The results of verification are shown in Table IV. An entry of YES indicates that the property has passed the verification while NO indicates otherwise. The blank entries indicate that the property is no relevant to the shaper. The value of the property *NI* in *TAS+Qbu* is NO, which means the preemption can not happen. It is same as the reason why the values of the property *RP* in *TAS* and *PS* are NO. Therefore, the results of properties 1-7 prove that our models follow the time-aware and peristaltic shaper's shaping rules.

By observing and analysing the diagnostic traces generated in the simulator of UPPAAL, we give the reasons why the property *LL* can not pass the verification in *TAS* and *PS* below. For **TAS**, if the real-time frames arrive at the start instant of BE time slots, they will be blocked for a long time. The waiting delay is too long for time-critical frames For **PS**, the time slot is very short. When best-effort frames are being transmitted, they will occupy the next time slot. In the worst case, the waiting delay of the real-time frames exceed two time slots, which is far greater than the delay requirement.

For **TAS+Qbu**, the preemption guarantees that the latency of real-time frames will meet the maximum delay requirement. Real-time frames do not need to wait for the length of BE time slot or the guard band. When the real-time frames arrive, they can be transmitted first. For **PS+Qbu**, real-time frames will not be interfered by best-effort frames with low-priority anymore. They can be sent immediately after sufficient residence time, so the waiting delay is much shorter. Fig.12 shows that the shapers with preemption mechanism can satisfy the time requirement.

D. Timing Analysis

By running and simulating on the shapers models we have established, we can obtain the recorded total time, the total transmission time, waiting delay and other variables. As shown

TABLE V
TIME PERFORMANCE ANALYSIS

Mechanism	T_{sum}	T_{global}	*Utilization*
TAS	$916\mu s$	$1000\mu s$	91.6%
	$1836\mu s$	$2000\mu s$	91.8%
	$2754\mu s$	$3000\mu s$	91.8%
	$3642\mu s$	$4000\mu s$	92.1%
TAS+Qbu	$976\mu s$	$1000\mu s$	97.6%
	$1956\mu s$	$2000\mu s$	97.8%
	$2943\mu s$	$3000\mu s$	98.1%
	$3928\mu s$	$4000\mu s$	98.2%

in Table V and VI, we analyze the time performance of experiments based on these data. T_{global} represents the value of global clock time after all frames have been transmitted. T_{sum} denotes the total transmission time of all frames. The resource

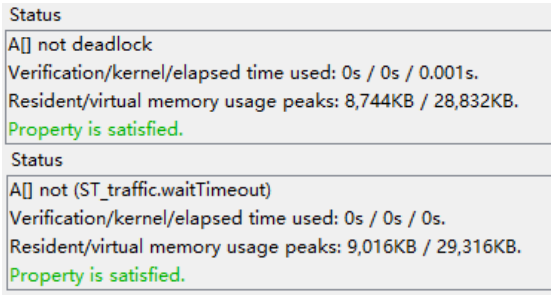

Fig. 12. Verification results.

utilization rate can be roughly calculated by the formula:

$$utilization = \frac{T_{sum}}{T_{global}} \times 100\%$$

In Table V, we calculate the resource utilization rate when $T_{global} = 1000, ..., 4000$. It shows that the resource utilization of TAS is much lower than that of TAS with preemption mechanism. Table VI shows the maximum delay time that an ST frame may have to wait. Experimental results show that when the payloads increase, the preemption mechanism can significantly improve the latency of time-critical frames and the resource utilization of the original protocol.

TABLE VI
DELAY ANALYSIS OF ST FRAMES

Mechanism	maxDelay/frame
TAS	250 μs
PS	40 μs
PS+Qbu	20 μs
TAS+Qbu	2 μs

V. CONCLUSION

Time-sensitive network is an effective solution to transmit real-time traffic in control systems. Meanwhile, the feasibility analysis of TSN is particularly important. Motivated by this, this paper presents formal models of the TSN traffic shapers. In order to evaluate whether these shapers are able to satisfy the strict timing requirements or not, we applied CTL formulas to describe the properties of the TSN protocol, i.e., deadlock-freedom, frame mutual exclusion,liveness of forwarding processes, non-interleaving, sequentiality of traffic, reachability of preemption and low latency. These properties could be verified by the verifier in UPPAAL. Besides, we also conducted a formal analysis of different shapers based on our models. In particular, we compared the resource utilization and the transmission latency of real-time frames with and without preemption. We found that the preemption mechanism can not only reduce the end-to-end latency but also significantly improve resource utilization. Through our models, designers and engineers can easily understand and develop the TSN better with higher reliability and hard real-time performance.

ACKNOWLEDGMENT

This work is partially supported by National Key Research and Development Program (2019YFB2102602).

REFERENCES

[1] C. E. Spurgeon, *Ethernet: the definitive guide*. " O'Reilly Media, Inc.", 2000.
[2] "Ieee standard for local and metropolitan area networks–bridges and bridged networks," 2014.
[3] I. of Electrical and E. E. (2019), "Time-sensitive networking task group. accessed: Jul. 6 [online]," 2016.
[4] G. Alderisi, A. Caltabiano, G. Vasta, G. Iannizzotto, T. Steinbach, and L. L. Bello, "Simulative assessments of ieee 802.1 ethernet avb and time-triggered ethernet for advanced driver assistance systems and in-car infotainment," in *2012 IEEE Vehicular Networking Conference (VNC)*, pp. 187–194, IEEE, 2012.
[5] E. M. Clarke Jr, O. Grumberg, D. Kroening, D. Peled, and H. Veith, *Model checking*. MIT press, 2018.
[6] G. Behrmann, A. David, and K. G. Larsen, "A tutorial on uppaal," in *Formal methods for the design of real-time systems*, pp. 200–236, Springer, 2004.
[7] K. G. Larsen and P. Pettersson, "Wang yi," *Uppaal in a nutshell. STTT*, vol. 1, no. 1-2, pp. 134–152, 1997.
[8] S. Saini and A. Fehnker, "Evaluating the stream control transmission protocol using uppaal," 2017.
[9] M. Kamali, "Modeling and verifying the olsr protocol using uppaal," 2014.
[10] J. Bengtsson and D. Griffioen, "Automated verification of an audio-control protocol using uppaal," *The Journal of Logic and Algebraic Programming*, vol. s 52–53, no. 1, pp. 163–181, 2002.
[11] L. L. Bello and W. Steiner, "A perspective on ieee time-sensitive networking for industrial communication and automation systems," *Proceedings of the IEEE*, vol. 107, no. 6, pp. 1094–1120, 2019.
[12] L. Zhao, P. Pop, Z. Zheng, and Q. Li, "Timing analysis of avb traffic in tsn networks using network calculus," in *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pp. 25–36, IEEE, 2018.
[13] P. Zhang, Y. Liu, J. Shi, Y. Huang, and Y. Zhao, "A feasibility analysis framework of time-sensitive networking using real-time calculus," *IEEE Access*, vol. 7, pp. 90069–90081, 2019.
[14] D. Hellmanns, J. Falk, A. Glavackij, R. Hummen, S. Kehrer, and F. Dürr, "On the performance of stream-based, class-based time-aware shaping and frame preemption in tsn," in *2020 IEEE International Conference on Industrial Technology (ICIT)*, pp. 298–303, IEEE, 2020.
[15] Y. Lu and M. Sun, "Modeling and verification of ieee 802.11 i security protocol for internet of things.," in *SEKE*, pp. 270–269, 2018.
[16] M. J. Teener, "Peristaltic shaper in clause 8 style," 2013.
[17] IEEE, "802.1qbu—frame preemption," 2015.