IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 30, 2019, accepted June 26, 2019, date of publication July 9, 2019, date of current version July 24, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2927516

A Feasibility Analysis Framework of Time-Sensitive Networking Using Real-Time Calculus

PENG ZHANG[1], YU LIU[1], JIANQI SHI[1,2], YANHONG HUANG[1,3], AND YONGXIN ZHAO[3]

[1]National Trusted Embedded Software Engineering Technology Research Center, East China Normal University, Shanghai 20062, China
[2]Hardware/Software Co-Design Technology and Application Engineering Research Center, East China Normal University, Shanghai 20062, China
[3]Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai 20062, China

Corresponding author: Jianqi Shi (jqshi@sei.ecnu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61602178, in part by the Shanghai Science and Technology Committee Rising-Star Program under Grant 18QB1402000, and in part by the Science and Technology Commission of Shanghai Municipality Project under Grant 18ZR1411600.

ABSTRACT In the field of intelligent connected vehicles, as the rate of in-vehicle communication continues to increase, the importance of real-time and reliability requirements has been more prominent. The time-sensitive networking (TSN) task group is dedicated to the amendments for meeting the urgent needs in the industrial field. In the implementation of the TSN, time latency bounds of critical traffic, memory bounds of nodes and utilization of transmission resources are needed to evaluate the feasibility of the TSNs. In this paper, we propose a framework called component-based analysis (CBA) to analyze these properties. In CBA, we use a data model, a resource model, and a component model to construct an abstract model of the TSNs with the system architecture and analyze the feasibility of this abstract model with real-time calculus. Moreover, we discuss the preemption mechanism mentioned in IEEE 802.1Qbu to handle the conflict of different priority queues, which has a serious influence on the resource model of the TSNs. Finally, we validate this framework by performing an analysis on a synthetic case and compare the obtained properties with different conflict handling mechanisms.

INDEX TERMS TSN, component-based analysis, real-time calculus, feasibility.

I. INTRODUCTION

With the integration of artificial intelligence, Internet of Things and the automotive industry, as well as the trend to integrate vehicles, vehicle-control and vehicle-cloud, the existing Internet of Vehicles has been unable to handle the increase in intelligent vehicle applications. To solve this problem, there is an urgent need for a high-speed and real-time system for intelligent networked automotive applications. With the continuous development of Internet technology, the transmission rate of Ethernet has been continuously improved from 10M, 100M to 10G. At the same time, the certainty and real-time requirement of the network have also become highly necessary [1].

Compared with traditional industries, intelligent connected vehicles have higher requirements for real-time communication. After years of exploration, some real-time network communication protocols, including the Real-time Transport Protocol [2] (RTP) and Audio Video Bridging [3] (AVB), have been developed, but the complexity of these protocols

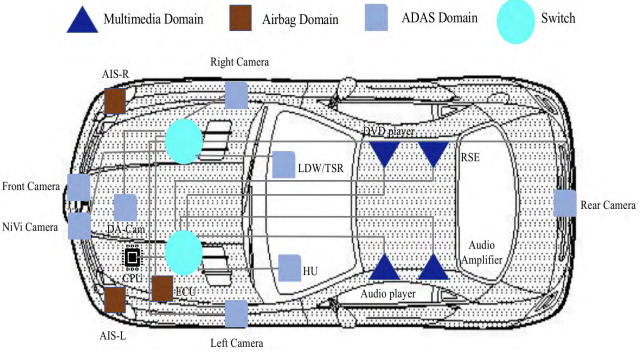

FIGURE 1. On-board network topology example.

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

 This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/

limits the generality and scalability of network communication, thereby limiting the development of the entire industry.

To promote the application of intelligent manufacturing, many manufacturers have joined to formulate a new real-time communication protocol standard — Time-Sensitive Networking (TSN) [4], which pushes industrial and automotive Ethernet into the Gbit/s era. The TSN task group is dedicated to researching Ethernet to meet the urgent needs with high real-time requirement and high reliability in the industrial field. TSNs guarantee real-time Ethernet communication through several substandards in many aspects, including clock synchronization, traffic scheduling, traffic monitoring, frame preemption mechanisms and redundancy mechanisms.

TSN is an excellent mechanism to realize real-time transmission through a global schedule configured by gate control lists (GCLs) indicated in IEEE 802.1Qbv [6]. In the implementation of TSN, time latency bounds of critical traffic and memory bounds of nodes are required to evaluate the TSNs. The latency bounds correspond to the real-time requirement of TSN. The memory bounds provide nodes with enough space to buffer the waiting traffic. These bounds directly reflect the feasibility of TSNs. Meanwhile, we use the utilization of transmission resources to estimate the TSNs where GCLs have been created in advance. A lower utilization of critical traffic means that abundant resources are available for uncritical traffic.

The main approach to analyze the bounds and utilization mentioned above is simulation-based techniques [7]. However, simulation-based techniques often require substantial effort to construct the TSN model with numerous details when we analyze the properties through simulations. Considerable time would be spent executing the models and obtaining the simulation results. Note that there will also be a substantial time consumption to create libraries of building blocks of simulation tools.The simulation-based analysis will clearly require a great deal of time and effort.

In this paper, we propose a framework called component-based analysis (CBA) to construct an abstract model of TSN, and we compute the hard upper and lower bounds using real-time calculus without the disadvantages described above. Our method only needs incoming and outgoing event rates, message sizes, execution routes and the GCLs to characterize the functionality of critical traffic and the capacity of processing. This method is more effective because the model is at a much higher level of abstraction than a typical simulation-based model.

In this work, we start from the assumption that the GCLs and the traffic class are given for each output port. We introduce CBA with real-time calculus. We construct the abstract model through CBA, and we calculate the latency bounds, memory bounds and utilization along the hops from senders to receivers by real-time calculus. We find that we can obtain different abstract models under different conflict handling mechanisms. Therefore, we discuss these mechanisms and present the bounds and utilization with different traffic conflict handling mechanisms.

The critical **contributions** of this study include the following three aspects:

1) We propose a CBA framework for feasibility analysis of TSNs. In CBA, we use the data model, the resource model and the component model to construct the abstract model of TSNs, and apply real-time calculus to analyze feasibility of the abstract model;
2) We discuss the resource model of TSNs under different conflict handling mechanisms. With introduction of the preemption mechanism mentioned in IEEE 802.1Qbu, the resource model will be more efficient when process the arrival traffic;
3) We use a synthetic case to validate the framework CBA and discuss the time bounds, memory bounds and resource utilization under different conflict handling mechanisms.

The remainder of this paper is organized as follows. In section II, we discuss the related works. In section III, we introduce the relevant portions of the IEEE 802.1Qbv mechanism, IEEE 802.1Qbu mechanism and the real-time calculus. We discuss the CBA framework for constructing an abstract model of TSNs in section IV. We present the detailed analysis of the constructed abstract models with real-time calculus in section V. In section VI, we use a case study to validate the framework that we proposed in this paper. Then, we present conclusion remarks in section VII. In the last part, we include an appendix to help readers better understand this paper.

II. RELATED WORK

The core function of a TSN is controlling the gate of each egress queue in the network system. A gate control list (GCL) is the core of IEEE 802.1Qbv that is used to control the forwarding of critical traffic. There are now several methods [8]–[10] to construct GCLs; these methods could be used to generate GCLs with high traffic transmission efficiency. In the process of GCL generation, the space of the solution and the speed of the solution are mutually constrained. For complex TSNs, perfect GCLs are difficult to obtain; thus, the GCLs we obtained will have some defects. Hence it is necessary to analyze the latency bounds, memory bounds and utilization of transmission resources.

Network-calculus is an efficient method to analyze the performance of complex networks. Network-calculus-based analysis was presented in [17], [20], [21] to compute the latency bounds of traffic in various networks. Reference [11] proposed a method to analyze the worst-case latency of TSNs for generic GCL tables. There is no need for us to consider the method for generating GCLs and the rationality of the GCLs; we only need to focus on the latency of the network. However, the method discussed in this paper only considers the calculation of time latency, which is a part of the consideration of network feasibility. And the TSN discussed in this paper does not consider the preemption mechanism proposed in IEEE 802.1Qbu.

The performance analysis of complex systems is a common problem in embedded systems. A system performance analysis framework was proposed in [13]. This framework can analyze the performance of the network system based on the system level, and it can analyze system performance, including latency, cache and bandwidth. Compared with other methods, the cost is significantly reduced, the analysis efficiency is clearly enhanced, and the verification result is sufficiently accurate. In [14], Wandeler *et al.* used the framework to model an embedded system and then evaluated the embedded system performance using the obtained model with real-time calculus.

In our study, we propose a component-based framework as a core method and apply it to mainly analyze the feasibility of TSN. Similar to [11], we need not to consider the GCLs; we focus the feasibility analysis under a given GCLs. To the best of our knowledge, this is the first systematic proposal for feasibility analysis of TSN. And compared to traditional simulation, it has a significantly faster speed and the results are accurate enough for performance analysis.

III. BACKGROUND

In this section, we provide a brief overview of TSN substandards IEEE802.1Qbv and IEEE802.1Qbu and a brief introduction to real-time calculus.

A. A BRIEF OVERVIEW OF TSN SUBSTANDARDS

TSN is a communication technology that extends from the field of video and audio data to the automotive field and is further extended to the industrial field. Here, we introduce the two substandards of TSN including IEEE802.1Qbv and IEEE802.1Qbu.

1) IEEE 802.1QbV

The core of TSN lies in the principle of time-triggered communication, which is standardized as IEEE 802.1Qbv. It provides latency and jitter guarantees for time-critical applications to satisfy real-time and deterministic requirements. The simplified internal representation of the switch is shown in Fig. 2. As shown in this figure, the switch has *n* input ports and 1 output port. Internally, the network traffic from different input ports first passes through the switch fabric and then through the priority filter. The purpose of the priority filter is to distribute the traffic frames from different ports to the corresponding priority queues. IEEE 802.1Q defines the specific format of VLAN frames and provides standards for bridged networks that can adapt to various application scenarios. According to the definition of Ethernet frames in IEEE 802.1Q, a priority filter determines which specific flow is forwarded to the corresponding priority queues based on the priority code point (PCP).

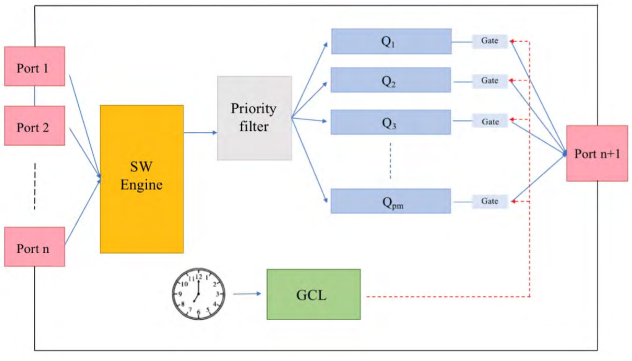

FIGURE 2. Representation of TSN queues and transmission gates.

Because TSN uses a time-triggered mechanism based on the different requirements of traffic transmission on latency and jitter, [8] proposed uniformly distributing traffic into scheduled traffic and nonscheduled traffic. Scheduled traffic (which has the highest priority by default) enters the scheduling queue with the corresponding priority through the priority filter, and nonscheduled traffic enters the remaining priority queue through the priority filter. Note that the scheduled traffic has a higher priority than all nonscheduled traffic, which means that the nonscheduled traffic will not affect the scheduled traffic. Each priority queue is followed by a corresponding gate structure that has two states of ''open'' and ''closed''. The state of the gate is configured by the GCL. Only when the state of the gate is ''open'' can network traffic be output through the switch. In addition, the entire network uses a time synchronization mechanism according to the IEEE 802.1ASrev standard to ensure time synchronization throughout the network. GCL strictly divides the communication period such that the scheduled traffic and the nonscheduled traffic can be transmitted in different time slots. Because the scheduled traffic is not interfered with by other traffic, the transmission of the scheduled traffic is guaranteed, thus ensuring the real-time and certainty requirements in the transmission process.

2) IEEE 802.1Qbu

IEEE 802.1Qbu also plays a very important role in the TSN standard cluster. The frame preemption mechanism mentioned in IEEE 802.1Qbu is similar to the preemption function in the operating system, that is, to suspend the currently nonurgent tasks and give priority to more urgent tasks. In a specific application scenario, when a low-priority frame is transmitted on a transmission line, if a high-priority frame wants to be transmitted preferentially because of higher time requirements, it will be judged at this time whether the low-priority frame meets the preemptive requirements, i.e., whether slicing is allowed. If the low-priority frame satisfies the conditions, the transmission of the low-priority frame will be terminated at the appropriate place while the transmission of the high-priority frame will be started. After the transmission of the high-priority frame is completed, the preamble is added in front of the remaining low-priority frames and encapsulated into the original frame, and then the transmission of the low-priority frames is continued. The preemptive mechanism can solve the delay of high-priority traffic transmission due to low-priority transmission, avoid

jitter in the transmission process, improve the real-time performance and reliability in the transmission process, and thus ensure the end-to-end transmission time. It cannot be ignored that although IEEE 802.1Qbu defines that low-priority frames can be preempted by high-priority frames, the preemption of Ethernet frames is limited, i.e., frames with a minimum of 127 bytes cannot be preempted. The preemption function principle is shown in Fig. 3.

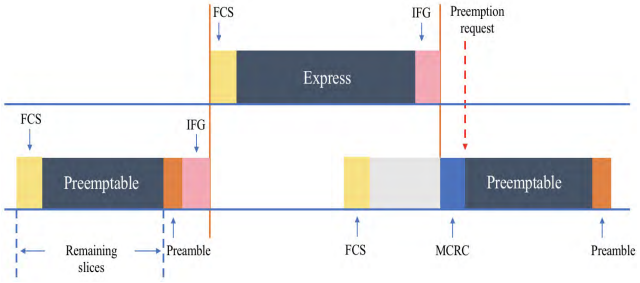

FIGURE 3. Simplified view of preemption.

B. REAL-TIME CALCULUS

Real-time calculus is based on the well-known network calculus, which is applied for various network analyses. The basis for real-time calculus is min-plus algebra. To analyze the system properties, we describe the node's traffic details through arrival curves and service curves. The basic concepts of min-plus algebra are given in Appendix A.

A cumulative function $R(t)$ is defined as the number of bits on the traffic in the time interval [0, t]. Function $R(t)$ is always increasing in a broad sense. By convention, we take $R(0) = 0$ unless otherwise specified.

Definition 1: Define an arrival curve tuple $\alpha(\Delta) = [\alpha^l(\Delta), \alpha^u(\Delta)]$ to describe the data model of a bit stream in a interval Δ from s to t, the relation of α^l, α^u, and R is presented by the following inequality:

$$\alpha^l(t-s) \le R(t) - R(s) \le \alpha^u(t-s), \quad \forall s < t. \tag{1}$$

with $\alpha^l(0) = \alpha^u(0) = 0$.

Similar to the cumulative function $R(t)$, a cumulative function $C(t)$ describes the available communication resource in the time interval [0, t]. Analogous to arrival curves, we provide a tuple $\beta(\Delta) = [\beta^l(\Delta), \beta^u(\Delta)]$ to constrain the cumulative function $C(t)$.

Definition 2: Define a service curve tuple $\beta(\Delta) = [\beta^l(\Delta), \beta^u(\Delta)]$ to describe the resource model for a component in a interval Δ from s to t, the relation of β^l, β^u, and C is presented by the following inequality:

$$\beta^l(t-s) \le C(t) - C(s) \le \beta^u(t-s), \quad \forall s < t. \tag{2}$$

with $\beta^l(0) = \beta^u(0) = 0$.

IV. COMPONENT-BASED ANALYSIS

In this section, we describe the details of our CBA framework to construct the abstract model of TSN. As shown in Fig. 4, we obtain the abstract model of TSN by constructing the resource model, data model and component model and then integrating them with the system architecture which is described by UML diagrams. It is convenient to analyze the abstract model using real-time calculus. In our framework, the data models are mainly used to describe the arriving data over time. The resource models describe the processing capacity of nodes that are available within a system. We provide component models to present nodes in a system. This component model describes the processing semantics that are used to process the arriving data with resources.

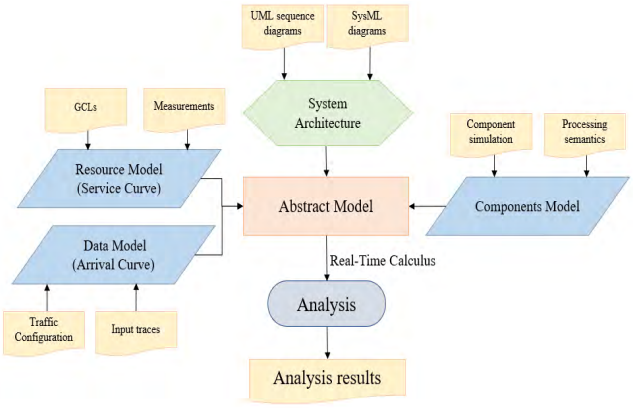

FIGURE 4. Framework of component-based analysis.

A. DATA MODEL (ARRIVAL CURVE)

The arrival curve is used to describe the data model of components, which represents the amount of bits reaching the component. We use the input traces and the traffic configuration to represent the information of data models. And a triple (p, j, d) can be obtained by these information, where p denotes the period of arriving flows, j is the jitter, and d is the minimum interval distance of traffic in the modeled stream. The lower and upper arrival curves are modeled by the following equations:

$$\alpha^l(\Delta) = \left\lfloor \frac{\Delta - j}{p} \right\rfloor. \tag{3}$$

$$\alpha^u(\Delta) = \min\{\left\lceil \frac{\Delta + j}{p} \right\rceil, \left\lceil \frac{\Delta}{d} \right\rceil\}. \tag{4}$$

Generally, d is much smaller than p, and Fig. 5 is a simple schematic diagram of the arrival curve.

B. RESOURCE MODEL (SERVICE CURVE)

Analogously to the data model, the concrete availability of a communication resource can be described by a service tuple $\beta(\Delta) = [\beta^u(\Delta), \beta^l(\Delta)]$ of upper and lower service curves. In CBA, we assume that the GCLs have been supplied and we construct the resource model with the GCLs.

In Fig. 6, we show an example to obtain the lower service curve. Time slots of the processing node are necessary for the next calculation. It is assumed that Q_{P_m} is the waiting queue

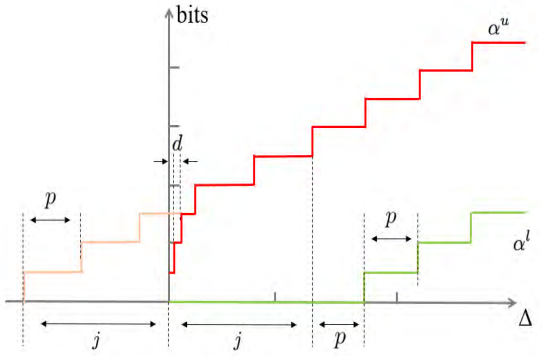

FIGURE 5. Arrival curves from period, jitter, and delay.

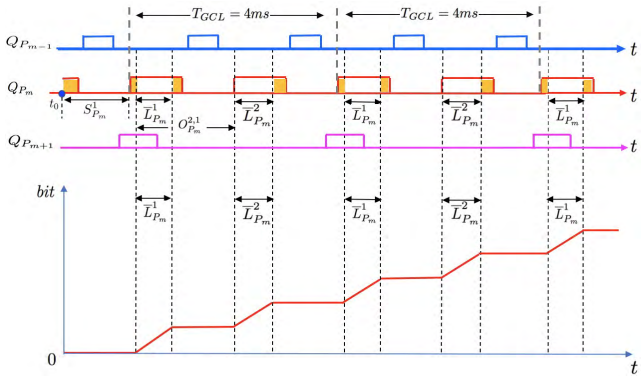

FIGURE 6. Worst-case service with guaranteed time slots for P_m.

of priority P_m. The priority of P_{m-} is higher than P_m, and P_{m+} is lower than P_m. As shown in Fig. 6, the open window of each queue in the TSN may not necessarily guarantee the forwarding of the priority traffic; it will be affected by the queue that overlaps with it. Meanwhile, it will also be affected by the lookahead mechanism. The lookahead mechanism indicates that when the window is opened, each traffic transmission will consider whether the traffic transmission can be completed in the remaining time slot. If the remaining time slot is not sufficient to complete the transmission, it will stop and wait for the next time slot. The length of the entire open window is L_{P_m}, and after considering the influence of other priorities and the lookahead mechanism on it, the guaranteed length of the open window is $\overline{L}_{P_m}$. In addition, S_{P_m} represents the longest waiting time, that is, the finish time of the last flow in the previous time window to the start time of the first flow in the next time window.

We regard the i-th time slot as the reference point of our calculation. The upper and lower service curves $\beta^{i,u}_{P_m}$ and $\beta^{i,u}_{P_m}$ for the critical traffic of priority P_m are given by

$$\beta^{i,u}_{P_m}(t) = \sum_{j=i}^{i+N_{P_m}-1} \beta^{j,i,u}_{P_m}(t). \tag{5}$$

$$\beta^{j,l}_{P_m}(t) = \sum_{j=i}^{i+N_{P_m}-1} \beta^{j,i,l}_{P_m}(t). \tag{6}$$

where

$$\beta^{j,i,u}_{P_m} = \beta_{T_{GCL},\overline{L}^j_{P_m}}(t + T_{GCL} - \overline{L}^j_{P_m} - O^{j,i}_{P_m}). \tag{7}$$

$$\beta^{j,i,l}_{P_m} = \beta_{T_{GCL},\overline{L}^j_{P_m}}(t + T_{GCL} - \overline{L}^j_{P_m} - S^i_{P_m} - O^{j,i}_{P_m}). \tag{8}$$

In Equations 7 and 8, T_{GCL} represents the *LCM* of the open-close cycles of critical traffic for all the queues, S^i_{Pm} denotes the longest waiting time before the i-th open window, and $O^{j,i}_{Pm}$ denotes the time interval between the opening times of the i-th open window and the j-th open window. N_{P_m} denotes the count of time slots in a GCL period. We calculate the service curve of one time slot, and the service curve of an entire GCL period is obtained by Equations 5 and 6. In addition, the service curve β is obtained by the following formula:

$$\beta_{T,L}(t) = C \cdot max(\left\lfloor \frac{t}{L} \right\rfloor L, t - \left\lceil \frac{t}{L} \right\rceil (T - L)). \tag{9}$$

Equations 5 and 6 are the service curve obtained on the i-th window, and Equations 10 and 11 show the final formula of the worst service curve due to slight differences caused by the reference slot. Finally, we obtain the synthesized service curve as shown in Fig. 6.

$$\beta^u_{Pm}(t) = \max_{1 \le i \le N_{Pm}} \{\beta^{i,u}_{Pm}(t)\}. \tag{10}$$

$$\beta^l_{Pm}(t) = \min_{1 \le i \le N_{Pm}} \{\beta^{i,l}_{Pm}(t)\}. \tag{11}$$

There are still several parameters above that we need to analyze combined with the actual *GCL* table, including $\overline{L}^i_{Pm}$, S^i_{Pm} and O^j_{Pm}. We find that the conflict handling mechanisms have a serious influence on these parameters.

Conflict handling mechanism

During the transmission of the Ethernet frames, the transmission must be guaranteed and cannot be interrupted by other frames. However, there will be uncertainties, such as the example in Fig. 6, where the last Ethernet frame has a conflict and infringes the high-priority time slot.

To ensure the transmission of critical traffic in time slots, a solution to address the conflict is proposed in [11], called the lookahead mechanism. It is noted that a guard band can alleviate these losses efficiently. With this solution, we check whether the remaining transmission time slot can satisfy the transmission of the flow before transmitting. If it can support the transmission, transfer it; otherwise, wait for the next cycle. There are two types of guard bands to protect high-priority time slots: flow guard bands and frame guard bands. During the flow guard band time, new critical flows cannot be transmitted, and only the flow that has not been completed can be transmitted. The length of the flow guard band must be long enough to enable the safe transmission of the critical flow before the next high-priority time slot. Therefore, the length of the flow guard band corresponds to the longest critical flow length. However, the flow guard band is too long to affect the regular transmission in each time slot. In contrast, the frame guard band can deal with

TABLE 1. Ethernet packet and frame structure.

Preamble	Start of frame delimiter	MAC destination	MAC source	802.1Q tag(optional)	Ethertype	Payload	32-bit CRC	Interpacket gap
7 octets	1 octet	6 octets	6 octets	(4 octets)	2 octets	46-1500 octets	4 octets	12 octets

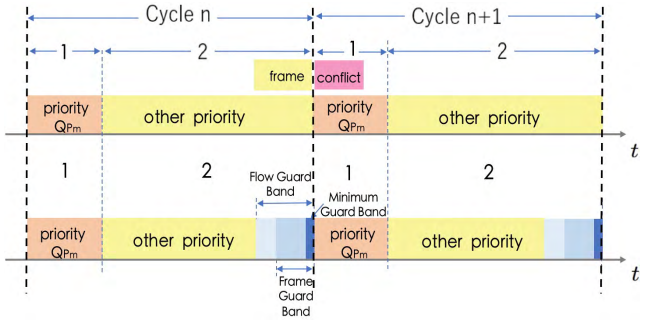

FIGURE 7. Guard bands prevent the infringement of time slot with critical traffic.

the relationship between the guard band and the guaranteed time slot better. The maximum length of the frame guard band equals the longest Ethernet frame length.

The format of the Ethernet frame is specified in IEEE 802.3, which consists of six parts: destination address, source address, 802.1Q tag (TPID and TCI), Ethertype, payload and frame check sequence. We can obtain the guard band length of Ethernet frame L_{gb}: 1500 byte (frame payload) + 18 byte (Ethernet addresses, Ethertype and CRC) + 4 byte (VLAN Tag) + 12 byte (interframe spacing) + 8 byte (preamble and SFD) = 1542 byte.

Although the presence of the guard band protects the transmission of high-priority time slots, it still has two significant defects.

1) In the guard band, no new traffic can be transmitted, and there is a serious waste of transmission resources.
2) Because the size of the time slice cannot be less than the length of the guard band, this greatly limits the size of the solution space of the GCL.

However, with the use of the lookahead mechanism only, the length of the guard band is not deterministic, and there is still a serious loss. To minimize the loss caused by the guard band, we use the preemption mechanism introduced in section II. At this point, the length of the guard band is deterministic and is completely dependent on the preemption mechanism. As shown in Table 1, the minimum Ethernet frame length is 64 bytes, so the length of the guard band L_{gb} can be reduced to 127 bytes: 64 byte (minimum frame) + 63 byte (remaining length that cannot be preempted).

In addition, the preemption mechanism plays a very important role when there is an overlap between different priority traffic. In traditional Ethernet, the transmission of high-priority traffic must wait until the transmission of low-priority traffic in the network is completed, which seriously increases the latency of critical traffic. Through the preemption mechanism, when critical traffic needs to be transmitted, low-priority traffic is ready to be truncated to facilitate the transmission of critical traffic immediately. In the worst case, the low-priority traffic just begins to transmit, and the length of the traffic does not meet the preemption requirement, whereby the traffic length is less than or equal to 127 bytes, which is similar to the calculation of the guard band.

The preemption mechanism mentioned in IEEE 802.1Qbu is an important method to minimize the loss of the guard band. The concrete details to obtain the necessary parameters, including $\overline{L}^{i}_{Pm}$, S^{i}_{Pm} and $O^{j,i}_{Pm}$, are described in Appendix B.

C. COMPONENT MODEL

In TSN, an incoming critical stream is typically processed on a sequence of components. For CBA with real-time calculus, we model such an abstract component as shown in Fig. 8. Here, an abstract data stream $\alpha(\Delta)$ enters the abstract component and is processed by an abstract resource $\beta(\Delta)$. The output is an abstract data stream $\alpha'(\Delta)$, and the remaining resources are expressed by $\beta'(\Delta)$.

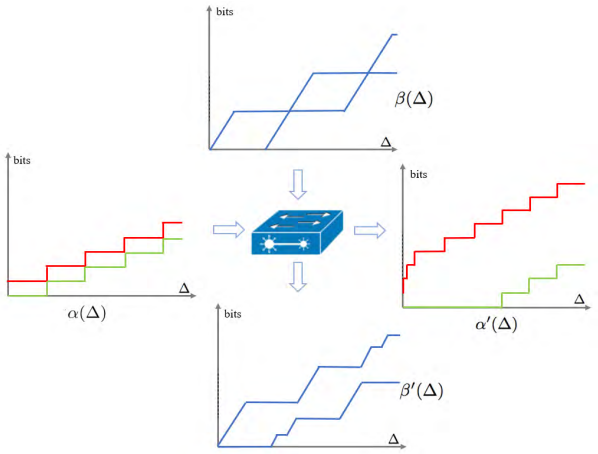

FIGURE 8. An abstract component, processing an abstract traffic on an abstract resource.

Internally, such an abstract component is specified by a set of functions that relate the incoming arrival curves and service curves to the outgoing arrival curves and service curves:

$$\alpha' = f_{\alpha}(\alpha, \beta). \tag{12}$$

$$\beta' = f_{\beta}(\alpha, \beta). \tag{13}$$

For a given abstract component, the relations f_{α} and f_{β} depend on the processing semantics of the modeled concrete component. In this work, we refer to the components as a

fixed priority (FP) components. And the components processing semantics are shown as following relations [13]:

$$\alpha'^{u} = min\{(\alpha^{u} \otimes^{*} \beta^{u}) \oslash \beta^{l}, \beta^{u}\}. \quad (14)$$

$$\alpha'^{l} = min\{(\alpha^{l} \oslash \beta^{u}) \otimes \beta^{l}, \beta^{l}\}. \quad (15)$$

$$\beta'^{u} = (\beta^{u} - \alpha^{l}) \overline{\oslash} 0. \quad (16)$$

$$\beta'^{l} = (\beta^{l} - \beta^{u}) \overline{\otimes} 0. \quad (17)$$

D. ABSTRACT MODEL OF TSN

At this point, we know how to model the data streams, the communication resources, and the components. However, to analyze the performance criteria of TSN, we need to build an abstract model of the complete system architecture.

For an individual node, its abstract model is an integration of the data model, resource model and component model. For TSN, the abstract model is composed of all abstract models of nodes on the basis of system architecture. It contains all the information that is necessary for feasibility analysis. By correctly interconnecting the inputs and outputs of all these abstract models, we obtain the abstract model of the system. An example of an abstract model of TSN is depicted in Fig. 9.

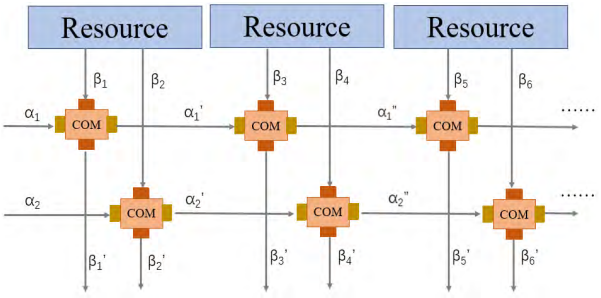

FIGURE 9. Abstract model of time-sensitive networking.

The input arrival curves and the processed arrival curves in the abstract model are interconnected to reflect the flow of data in the system horizontally, while the interconnections of the input service curves and the remaining service curves are presented in the system vertically. In TSN, components are allocated to the communication resource according to the GCLs of TSN. It is clear that the allocation of resources is closely related to GCLs. However, the GCLs of TSN do not guarantee that the resource has no conflict between the two components. Therefore, we discuss two conditions: 1) GCLs are excellent enough and there is no overlap between two time slots, and 2) there is overlap between two time slots. In the following, we describe these two conditions through an example.

We now describe how to model the abstract model in detail. In Fig. 10, we have two input flows, passing through two switches and finally reaching the target end system. We define

*$\otimes$ and $\oslash$ represent the convolution and deconvolution of min-plus calculi. $\overline{\otimes}$ and $\overline{\oslash}$ represent the convolution and deconvolution of max-plus calculi. We introduce min-plus and max-plus calculi in appendix A.

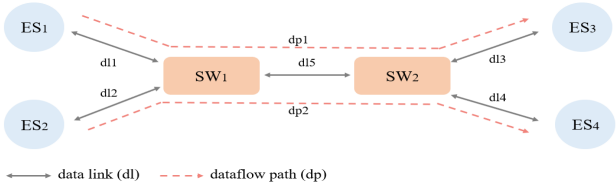

FIGURE 10. TSN cluster example.

the traffic of the end systems ES_1 and ES_2 as S_1 and S_2, and the corresponding priorities are P_1 and P_2, respectively. Note that the priority of P_1 is higher than that of P_2.

Condition 1: It is assumed that the GCLs of TSN are in an ideal state, that is, the open windows of Q_1 and Q_2 in the GCL table have no overlap, and each traffic is transmitted in its own open window without mutual interference. In Fig. 11, SW_1 schedules S_1 and S_2 according to its GCLs. Subsequently, the two processed flows S_1' and S_2' enter SW_2, and SW_2 forwards the traffic according to its GCLs in the same way. Finally, we obtain the processed flows S_1'' and S_2''.

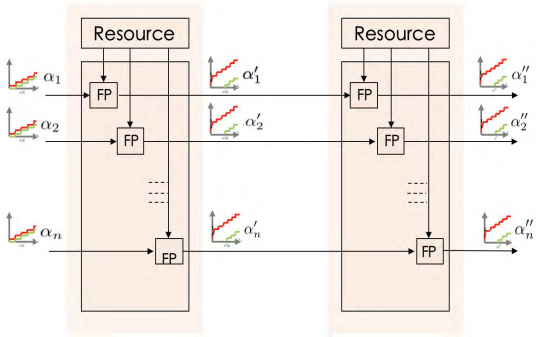

FIGURE 11. Simplified model of a scheduling network.

Condition 2: It was noted in the previous discussion that GCLs that have no overlap for a complex network are substantially nonexistent. For each switch, if the time slot of queues Q_1 and Q_2 overlap, streams S_1 and S_2 are the relationships of competing transmission resources. S_1 has a higher priority than S_2, and the model is shown in Fig. 12 when they compete for resources.

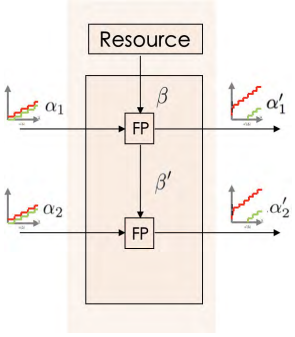

FIGURE 12. Simplified model of a scheduling network with overlap.

Assume that the gates of the two buffer queues Q_1 and Q_2 are simultaneously opened, and the resource will first serve the arriving traffic α_1, which belongs to the high-priority queue Q_1, and the remaining resource processed β' will process α_2. The latest resource will be used on the event-triggered traffic. Condition 2 is the case when the corresponding gates of each arriving flow are simultaneously opened. In the worst case, the resource β will be used up by α_1, and the remaining traffic will be blocked in the buffer queue until the next cycle.

Through the above method, we can obtain the abstract model of TSN. Then, the performance of the network can be obtained by the method presented in section V.

V. PERFORMANCE ANALYSIS

After interconnecting all abstract models of a system to the system's abstract model as described in the previous section, this abstract model captures all the information for performance analysis with real-time calculus.

For every node in the network, α^u and α^l represent the upper and lower arrival curves of a given data stream, and $\alpha^{u'}$ and $\alpha^{l'}$ denote the processed arrival curves. Similarly, β^u and β^l represent the upper and lower service curves, and $\beta^{u'}$ and $\beta^{l'}$ denote the remaining service curves of the resource. These curves are related by the processing semantics of the component model as Equations 14 to 17. With these relationships, we can obtain the arrival curve and service curve of each component.

Through the data model and the resource model obtained above, we can build an abstract model of the node, which contains various performance information. With the implementation of calculus theory, we analyze the latency bounds of critical traffic and the memory bounds of components, as shown in Fig. 13. We can visually observe that the maximum delay is the maximum horizontal distance between the service curve and the arrival curve, and the maximum buffer is the maximum vertical distance between the service curve and the arrival curve.

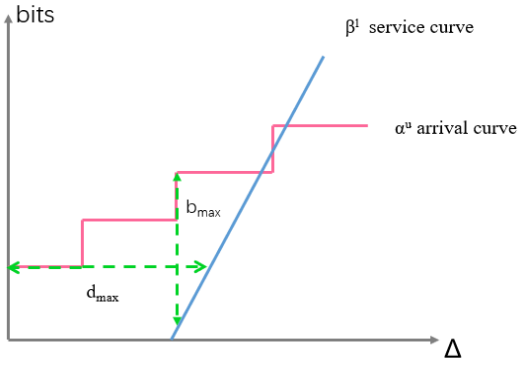

FIGURE 13. Abstract representation of delay and backlog.

When critical traffic with arrival curve α is processed by a component on a resource model with service curve β, the worst-case latency delay is bounded by:

$$delay \leq \sup_{t\geq 0}\{\inf\{\tau \geq 0 : \alpha^u(t) \leq \beta^l(t+\tau)\}\} \stackrel{def}{=} Del(\alpha^u, \beta^l). \quad (18)$$

Pay bursts only once:

In a concatenation of TSN nodes, the method that calculates the performance is in a pessimistic situation. However, critical traffic is generally periodic, and bursts must be processed only once. Without the mechanism, the critical traffic will be processed by a series of components in TSN. We can obtain the delay requirements of the whole system through simple addition:

$$d_{max} \leq \sum_{t=1}^{n} Del(\alpha_t^u, \beta_t^l). \quad (19)$$

When we focus on the "pay bursts only once" phenomenon, the end-to-end delay is tightened by:

$$d_{max} \leq Del(\alpha^u, \beta_1^l \otimes \beta_2^l \otimes \ldots \otimes \beta_n^l). \quad (20)$$

Moreover, it is necessary to obtain the minimum memory requirement for each node to prevent overflow. We can obtain the memory bound by:

$$memory \leq \sup_{t>0}\{\alpha^u(t) - \beta^l(t)\} \stackrel{def}{=} Buf(\alpha^u, \beta^l). \quad (21)$$

Similarly, due to the "pay bursts only once" phenomenon, the actual memory requirement for a component is tightened by:

$$m_{max} \leq Buf(\alpha^u, \beta_1^l \otimes \beta_2^l \otimes \ldots \otimes \beta_n^l). \quad (22)$$

In addition, the utilization of node resources is also an important criterion for evaluating the quality of TSN. In the previous chapters, we already obtained β^u and $\beta^{l'}$, and the maximum utilization is given by:

$$utilization = \lim_{\Delta\to\infty} \frac{\beta^u(\Delta) - \beta^{l'}(\Delta)}{\beta^u(\Delta)}. \quad (23)$$

With the use of utilization, we can also analyze the potential bottlenecks of TSN.

VI. CASE STUDY

In this section, we evaluate the efficiency of our proposed framework for the performance analysis of TSN through a realistic vehicle case study. The impact of the abstract model on the system feasibility is discussed herein with reference to the simulation results.

As illustrated in Fig. 14, our case study is a representative vehicle sensor/actuator network based on TSN at 1 Gbit/s interconnecting four end systems and two switches that support TSN. As mentioned, we assume that the GCLs for each priority queue have already been obtained. The GCL and critical flows that we use are given in Tables 2 and 3. We give

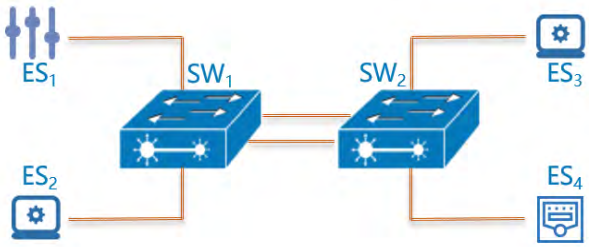

FIGURE 14. Time-sensitive networking architecture.

TABLE 2. GCLs involved in the case. Assume the period of each time slot is 200μs.

Link	Priority	Time Slot (μs)
$[ES_1, SW_1]$	8	[30, 50]
	6	[45, 65]
	5	[100, 120]
	4	[125, 145]
	3	[85, 105]
$[ES_2, SW_1]$	7	[40, 60]
	5	[150, 170]
	4	[25, 45]
	3	[175, 195]
	2	[95, 115]
	1	[80, 100]
$[SW_1, SW_2]$	8	[40, 60]
	7	[150, 170]
	6	[135, 155]
	5	[20, 40]
	4	[170, 190]
	3	[95, 115]
	2	[75, 95]
	1	[60, 80]
$[SW_2, ES_3]$	6	[30, 50]
	5	[150, 170]
	4	[90, 110]
	3	[60, 80]
	2	[15, 35]
	1	[135, 155]
$[SW_2, ES_4]$	7	[105, 125]
	6	[40, 60]
	5	[125, 145]
	4	[150, 170]

the GCLs of the priority queue in each pair of transmissions in the network, where each time slot will be affected by other priority time slots. The table on critical flows also lists all the necessary information.

The performance simulation process is performed as follows. We first obtain the resource model for each priority queue of each component based on the obtained GCLs.

TABLE 3. Configurations and parameters of time-triggered traffic.

Flow	Priority	Route	Period (μs)	Size (B)
TT_1	6	ES_1-SW_1-SW_2-ES_3	200	300
TT_2	4	ES_1-SW_1-SW_2-ES_3	200	440
TT_3	3	ES_1-SW_1-SW_2-ES_3	100	140
TT_4	2	ES_2-SW_1-SW_2-ES_3	200	390
TT_5	1	ES_2-SW_1-SW_2-ES_3	400	490
TT_6	4	ES_1-SW_1-SW_2-ES_4	400	560
TT_7	6	ES_1-SW_1-SW_2-ES_4	200	600
TT_8	4	ES_2-SW_1-SW_2-ES_4	200	500
TT_9	5	ES_2-SW_1-SW_2-ES_4	100	210
TT_{10}	7	ES_2-SW_1-SW_2-ES_4	200	370

Three factors will be considered here: i) the impact of the low-priority queue; ii) the impact of the high-priority queue; and iii) the impact of the lookahead mechanism. It was noted in section IV that three types of guard bands, flow guard bands, frame guard bands and minimum guard bands, will be considered in the resource model. Combining these influencing factors, we can obtain an accurate service curve of the resource model. The construction of the data model was discussed in section IV, and we obtain the arrival curve based on the information of the critical flow. Next, we can generalize the abstract model based on the components model with the TSN architecture mentioned in IV.

Combined with the arrival curve of the data model and the service curve of the resource model, we can obtain the worst-case latency of the TSN quickly. As mentioned previously, the worst-case latency is the horizontal maximum distance between the arrival curve and the service curve. However, different guard bands can lead to different resource curves. We will discuss the worst-case latency with flow guard band, frame guard band and minimum guard band. The obtained latency under three types of guard bands is illustrated in Fig. 15.

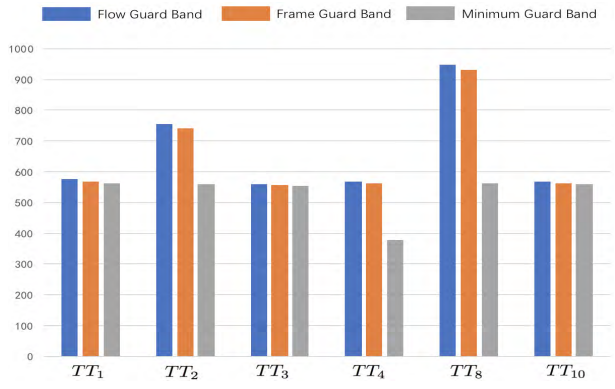

FIGURE 15. Worst-case latency with different guard band mechanisms.

TABLE 4. Worst-case backlog of processing node.

Node	WCB (bits)
ES_1	17360
ES_2	17440
SW_1	69600
SW_2	87920

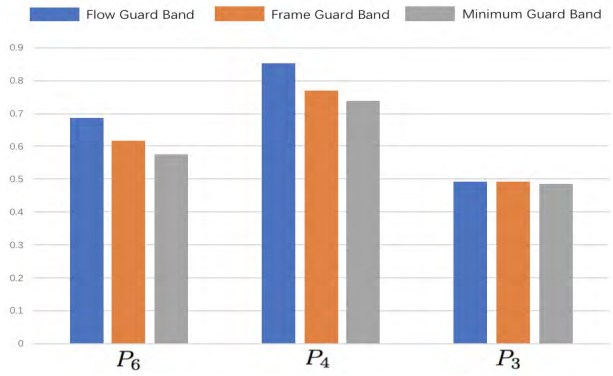

FIGURE 16. Maximum utilization with different guard band mechanisms.

We choose several flows to calculate their specific worst-case latency over the three guard band mechanisms. Through the comparison in Fig. 15, it is clear that the worst-case latency decreases in the three cases of flow guard band, frame guard band and minimum guard band. The minimum guard band is the most effective mechanism.

During the transmission of a TSN, each node's priority queue needs enough memory to store the waiting critical traffic. The memory corresponds to the maximum vertical distance of the arrival curve and the service curve. Through our analysis framework, we can calculate the minimum required memory for each priority queue of each node in this case study. However, our environment model uses a ladder model, which causes the influence of different types of protection bands on the nodes to be negligible. We present the concrete minimum required buffer size in this case study in Table 4.

In addition, we can analyze the resource utilization of each connection. Here, we provide an example of the time slots of the ES_1-SW_1 connection in the TSN. The utilization rate is expressed according to the maximum resource curve and the processed minimum resource curve of each connection, and the obtained result corresponds to the maximum resource utilization.

In this case of ES_1, only the time slots of the P_3, P_4, and P_6 priority queues are used to transmit traffic. Therefore, the utilization of the resource of other priority queues is 0. The maximum resource utilization for different priority queues is listed in Fig. 16. It is clear that the minimum guard band costs the least amount of resources when transmitting the same traffic, which is the more efficient mechanism in processing traffic.

VII. CONCLUSION

TSNs are an effective solution to transmit critical traffic in the Internet of Vehicles and are the focus of R&D by various vendors today. Meanwhile, the feasibility analysis of TSNs is particularly important. Motivated by this, this paper proposes a feasibility analysis solution for TSNs. We propose a framework called CBA to model the specified network and then analyze the constructed model using real-time calculus.

In addition, it is noted that different guard band mechanisms can lead to different performance. The preemption mechanism proposed in IEEE 802.Qbu is better to process the conflict in a node.

We use a synthetic case to validate our proposed CBA framework. It is proven that CBA can quickly analyze the feasibility of a TSN and obtain sufficiently accurate results. This framework can play an important role in quickly determining the feasibility of the network architecture and can effectively save costs in the design phase. And a comparison of feasibility analysis based on an actual TSN and the analyzing results of proposed framework in this paper is a subject for future work.

APPENDIX A
MIN-PLUS AND MAX-PLUS CALCULI

Today, real-time calculus is used increasingly in the fields of network modeling and quantitative analysis. The basis of real-time calculus is min-plus and max-plus algebra. With the introduction of real-time calculus into the field of network analysis, min-plus and max-plus algebra have made many significant achievements while improving their own theory and gradually evolving into itself. This makes real-time calculus evolve into another relatively independent system.

Min-plus algebra is a type of algebraic structure, denoted as $(R \cup \{+\infty\}, \wedge, +)$, where R represents the real set, $\wedge$ represents the minimum operation ($a \wedge b = min\{a, b\}$), and $+$ is the addition operation.

In the algebraic structure $(R, +, \times)$, the integral of the function is expressed as $\int_0^t f(s)$. Following the integral definition in the algebraic structure $(R, +, \times)$, in min-plus algebra $(R \cup \{+\infty\}, \wedge, +)$, the 'addition' corresponds to $\wedge$, and the 'multiplication' corresponds to $+$, so the 'integral' of the function f is defined as follows:

$$\inf_{s \in R \cup 0 \le s \le q} \{f(s)\}. \tag{24}$$

In min-plus and max-plus algebra, convolution and deconvolution are commonly used operations. These operations are defined as follows.

Definition 3: The min-plus convolution $\otimes$ and deconvolution $\oslash$ of f and g are defined as:

$$(f \otimes g)(t) = \inf_{0 \le s \le t} \{f(t-s) + g(s)\}. \tag{25}$$

$$(f \oslash g)(t) = \sup_{s \ge 0} \{f(t+s) - g(s)\}. \tag{26}$$

The max-plus deconvolution is similar to the min-plus convolution.

Definition 4: The max-plus convolution $\overline{\otimes}$ and deconvolution $\overline{\oslash}$ of f and g are defined as:

$$(f\overline{\otimes}g)(t) = \sup_{0\le u\le t}\{f(t-u)+g(u)\}. \tag{27}$$

$$(f\overline{\oslash}g)(t) = \inf_{u\ge 0}\{f(t+u)-g(u)\}. \tag{28}$$

For more information on min-plus calculi, see [15].

APPENDIX B
$\overline{L}_{P_m}$, $O^{j,i}_{P_m}$ AND $S^i_{P_m}$

For the guaranteed window size L of the waiting queue, it is affected by other priority queues and the lookahead mechanism. An example of the effect has been given in Fig. 6. The opening time and the closing time of the i-th window are denoted by $t^{o,i}_{P_m}$ and $t^{c,i}_{P_m}$, respectively. We discuss the size of the guaranteed window from three perspectives.

A. LOWER PRIORITY TRAFFIC

When the window in the low-priority queue has not finished yet, and the window of the high-priority queue is already open, the low-priority traffic needs to be completed, and then the high-priority traffic can be transmitted. The non-preemption delays caused by all lower-priority queues Q_{m^+} are:

$$d^{np,i}_{P_{m^+}} = \begin{cases} min\{\frac{l^{max}_{P_{m^+}}}{C}, t^{c,i}_{P_{m^+}} - t^{o,i}_{Pm}\}, & G_{P_{m^+}}(t^{o,i}_{P_m}) = 1 \\ 0, & G_{P_{m^+}}(t^{o,i}_{P_m}) = 0 \end{cases} \tag{29}$$

where

$$t^{c,i}_{P_{m^+}} = \inf_{t\ge t^{o,i}_{P_m}}\{G_{P_{m^+}}(t)=0\}. \tag{30}$$

$t^{c,i}_{P_{m^+}}$ denotes the nearest time when the gate $G_{P_{m^+}}$ is closed no earlier than $t^{o,i}_{Pm}$. The open time of the guaranteed time window after being affected by the lower-priority queue is:

$$t^{np,i}_L = d^{np,i}_L + t^{o,i}_{P_m}. \tag{31}$$

B. LOOKAHEAD MECHANISM

The lookahead mechanism means that when the gate is opened, each time the traffic is transmitted, it is considered whether the remaining time slot can guarantee the completion of the transmission. If the remaining time slot is not enough, the transmission will stop and wait for the next time slot to arrive. After the considering of the preemption mechanism, the length of the guard band can be reduced to 127 bits at the end of each transmission time slot.

The end time of the open window caused by the lookahead mechanism becomes:

$$t^{gb,i}_{P_m} = t^{c,i}_{P_m} - d^{gb}_{P_m}. \tag{32}$$

where $d^{gb}_{P_m}$ is the length of guard band which correspond the transmission capacity:

$$d^{gb}_{P_m} = \frac{l^{max}_{P_m}}{C}. \tag{33}$$

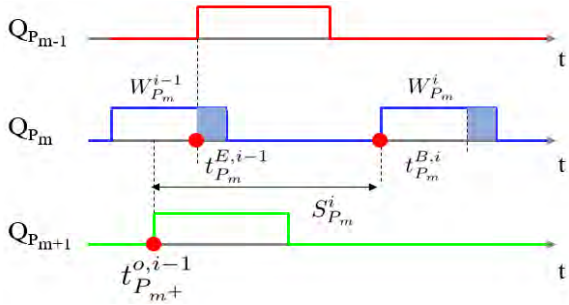

FIGURE 17. The maximum waiting time at the beginning busy period.

C. HIGHER PRIORITY TRAFFIC

Since the higher-priority queue is absolutely preemptive for the lower-priority queue, for the worst case performance analysis, the change of begin and end time of the open window caused by the high priority queue is:

$$t^{B,i}_H = \max_{1\le m^-\le m-1}\{t^{c,i}_{P_{m^-}}\}. \tag{34}$$

$$t^{E,i}_H = \min_{1\le m^-\le m-1}\{t^{o,i}_{P_{m^-}}\}. \tag{35}$$

where

$$t^{c,i}_{P_{m^-}} = \inf_{t\ge t^{o,i}_{Pm}}\{G_{P_{m^-}}(t)=0\}. \tag{36}$$

$$t^{o,i}_{P_{m^-}} = \sup_{t\le t^{c,i}_{Pm}}\{G_{P_{m^-}}(t)=0\}. \tag{37}$$

D. COMPREHENSIVE

$$t^{B,i}_{P_m} = \max\{t^{np,i}_L, t^{B,i}_H\}. \tag{38}$$

$$t^{E,i}_{P_m} = \min\{t^{gb,i}_{P_m}, t^{E,i}_H\}. \tag{39}$$

When the traffic has started to transmit, it must be completed. The guaranteed length of time slot is:

$$\overline{L}^i_{P_m} = \begin{cases} max\{t^{E,i}_{P_m} - t^{B,i}_{P_m}, \frac{l^{min}_{P_m}}{C}\}, & t^{B,i}_{P_m} \le t^{E,i}_{P_m} \\ 0, & t^{B,i}_{P_m} \ge t^{E,i}_{P_m} \end{cases} \tag{40}$$

where $l^{min}_{P_m} = 127$ when the preemption mechanism is considered.

And then it is necessary to calculate $O^{j,i}_{P_m}$ and $S^i_{P_m}$. $O^{j,i}_{P_m}$ means the time interval between the real opening time of the i-th open window and the j-th open window.

$$O^{j,i}_{P_m} = (j-i)\cdot T_{P_m} + \overline{O}^j_{P_m} - \overline{O}^i_{P_m}. \tag{41}$$

where

$$\overline{O}^i_{P_m} = t^{B,i}_{P_m} - t^{o,i}_{P_m}|_{\overline{L}^i_{P_m}\neq 0}. \tag{42}$$

$S^i_{P_m}$ denotes the longest waiting time before the i-th open window, but it is not a simple difference between $t^{B,i}_{P_m}$ and $t^{E,i-1}_{P_m}$. When it has an overlap with the low priority queue, the size of $S^i_{P_m}$ will be affected by the low priority queue

because of the transmission of low priority traffic in the end phase.

The latency $d_L^{np,0}$ produced by the lower priority traffic here is:

$$d_L^{np,0} = \max_{m+1 \le m^+ \le n} \{\min\{\frac{l_{P_{m^+}}^{max}}{C}, t_{P_m}^{E,i-1} - t_{P_{m^+}}^{o}\}\}. \quad (43)$$

The maximum latency is also affected by the preemption mechanism where $l_{P_{m^+}}^{\max}$ equals to 127 bytes. We can obtain the longest waiting time:

$$S_{P_m}^i = d_L^{np,0} + t_{P_m}^{B,i} - t_{P_m}^{E,i-1}. \quad (44)$$

For more information on min-plus calculi see [11].

REFERENCES

[1] C. E. Spurgeon, *Ethernet: The Definitive Guide*. Newton, MA, USA: O'Reilly Media, 2000.
[2] R. Frederick, V. Jacobson, and P. Design, *RTP: A Transport Protocol for Real-Time Applications*, document IETF RFC 3550, 2003.
[3] J. Imtiaz, J. Jasperneite, and L. Han, "A performance study of Ethernet audio video bridging (AVB) for industrial real-time communication," in *Proc. IEEE Conf. Emerg. Technol. Factory Automat.*, Sep. 2009, pp. 1–8.
[4] Institute of Electrical and Electronics Engineers. (2019). *Time-Sensitive Networking Task Group*. Accessed: Jul. 6, 2016. [Online]. Available: http://www.ieee802.org/1/pages/tsn.html
[5] Institute of Electrical and Electronics Engineers. (2019). *802.1AS-Rev—Timing and Synchronization for Time-Sensitive Applications*. [Online]. Available: https://1.ieee802.org/tsn/802-1as-rev/
[6] Institute of Electrical and Electronics Engineers. (2019). *802.1Qbv—Enhancements for Scheduled Traffic*. [Online]. Available: http://www.ieee802.org/1/pages/802.1bv.html
[7] G. F. Riley and T. R. Henderson, "The NS-3 network simulator," in *Modeling and Tools for Network Simulation*. Berlin, Germany: Springer, 2010, pp. 15–34.
[8] S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner, "Scheduling real-time communication in IEEE 802.1Qbv time sensitive networks," in *Proc. 24th Int. Conf. Real-Time Netw. Syst. (RTNS)*, Oct. 2016, pp. 183–192.
[9] P. Pop, M. L. Raagaard, S. S. Craciunas, and W. Steiner, "Design optimisation of cyber-physical distributed systems using IEEE time-sensitive networks," *IET Cyber-Phys. Syst., Theory Appl.*, vol. 1, no. 1, pp. 86–94, 2016.
[10] R. S. Oliver, S. S. Craciunas, and W. Steiner, "IEEE 802.1 Qbv gate control list synthesis using array theory encoding," in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS)*, Apr. 2018, pp. 13–24.
[11] Z. Luxi, P. Paul, and S. S. Craciunas, "Worst-case latency analysis for IEEE 802.1 Qbv time sensitive networks using network calculus," *IEEE Access*, vol. 6, pp. 41803–41815, 2018.
[12] Institute of Electrical and Electronics Engineers. (2019). *802.1Qbu—Frame Preemption*. [Online]. Available: http://www.ieee802.org/1/pages/802.1bu.html
[13] S. Chakraborty, S. Künzli, and L. Thiele, "A general framework for analysing system properties in platform-based embedded system designs," in *Proc. Design, Automat. Test Eur. Conf. Exhib.*, vol. 3, 2003, pp. 190–195.
[14] E. Wandeler, L. Thiele, M. Verhoef, and P. Lieverse, "System architecture evaluation using modular performance analysis: A case study," *Int. J. Softw. Tools Technol. Transf.*, vol. 8, no. 6, pp. 649–667, 2006.
[15] J.-Y. Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer, 2001.
[16] D. D. Khanh and A. Mifdaoui, "Timing analysis of TDMA-based networks using network calculus and integer linear programming," in *Proc. IEEE Int. Symp. Modelling, Anal. Simulation Comput. Telecommun. Syst.*, Sep. 2014, pp. 21–30.
[17] L. Zhao, P. Pop, Q. Li, J. Chen, and H. Xiong, "Timing analysis of rate-constrained traffic in TTEthernet using network calculus," *Real-Time Syst.*, vol. 53, no. 2, pp. 254–287, 2017.
[18] D. Shrestha, Z. Pang, and D. Dzung, "Precise clock synchronization in high performance wireless communication for time sensitive networking," *IEEE Access*, vol. 6, pp. 8944–8953, 2018.
[19] N. G. Nayak, "Scheduling & routing time-triggered traffic in time-sensitive networks," OPUS, Tech. Rep., 2018.
[20] L. Zhao, H. Xiong, Z. Zheng, and Q. Li, "Improving worst-case latency analysis for rate-constrained traffic in the time-triggered Ethernet network," *IEEE Commun. Lett.*, vol. 18, no. 11, pp. 1927–1930, Nov. 2014.
[21] M. Boyer, H. Daigmorte, N. Navet, and J. Migge, "Performance impact of the interactions between time-triggered and rate-constrained transmissions in TTEthernet," in *Proc. Eur. Congr. Embedded Real Time Softw. Syst.*, 2016, pp. 1–10.

PENG ZHANG was born in Chongqing, China, in 1995. He received the B.S. degree in electronic engineering from East China Normal University, Shanghai, China, in 2017, where he is currently pursuing the M.S. degree in software engineering. Since 2017, he has been a Research Assistant with the National Trusted Embedded Software Engineering Technology Research Center. His research interests include industrial safety, industrial networks, and model checking.

YU LIU was born in Huangshan, Anhui, China, in 1994. She received the B.S. degree in computer science and technology from Hefei Normal University, Hefei, China, in 2016. She is currently pursuing the M.S. degree in software engineering with East China Normal University, Shanghai, China. Since 2017, she has been a Research Assistant with the National Trusted Embedded Software Engineering Technology Research Center. Her research interests include industrial safety, industrial networks, and formal verification.

JIANQI SHI was born in Tianjin, 1984. He received the B.S. degree in software engineering and the Ph.D. degree in computer science from East China Normal University, Shanghai, China, in 2007 and 2012, respectively.

From 2012 to 2014, he was a Researcher Fellow with the National University of Singapore. From 2014 to 2014, he was a Research Scientist with the Temasek Laboratory under the Ministry of Defense of Singapore. He is currently an Associate Researcher with the School of Computer Science and Software Engineering, East China Normal University. His research interests include formal method, formal modeling, and the verification of real-time or control systems, and IEC 61508, IEC 61131 standards.

His awards and honors include ACMCCF Nomination of Excellent Doctor in Shanghai, in 2014 and Shanghai Science and Technology Committee Rising-Star Program, in 2018.

YANHONG HUANG was born in Neijiang, Sichuan, 1986. She received the B.S. degree in software engineering and the Ph.D. degree in computer science from East China Normal University, Shanghai, China, in 2009 and 2014, respectively. She has been with the School of Computer Science and Software Engineering, East China Normal University, as an Assistant Researcher. Since 2012, she has also been a Research Student with Teesside University, U.K. Her research interests include formal method, semantics theory, analysis, and the verification of embedded systems and industry software.

She was a recipient of the National Scholarship, in 2013, the IBM China Excellent Students, in 2013, and the Shanghai Excellent Graduates, in 2009 and 2014.

YONGXIN ZHAO was born in Mianyang, Sichuan, China, in 1983. He received the B.S. degree in mathematics and the Ph.D. degree in computer science from East China Normal University, Shanghai, China, in 2007 and 2012, respectively. He is currently an Associate Professor with the School of Computer Science and Software Engineering, East China Normal University. He held a postdoctoral position with the School of Computing, National University of Singapore, Singapore, from 2012 to 2014. He has more than 43 refereed publications. His research interests include program analysis and verification, semantics theory, web services, and formal methods.