Electronic Preprint for Journal of Information Processing Vol.20 No.2

Regular Paper

The Worst-Case Response Time Analysis for FIFO-based Offset Assigned CAN Messages

Yang Chen[1,a)] Ryo Kurachi[2] Gang Zeng[3] Hiroaki Takada[1]

Received: June 20, 2011, Accepted: December 16, 2011

Abstract: The Controller Area Network (CAN) is widely employed in automotive control system networks. In the last few years, the amount of data has been increasing rapidly in these networks. With the purpose of improving CAN bandwidth efficiency, scheduling and analysis are considered to be particularly important. As an effective method, it has been known that assigning offsets to the CAN messages can reduce their worst case response time (WCRT). Meanwhile, the fact is that many commercial CAN controllers have been equipped with a priority or first-in-first-out (FIFO) queue to transmit messages to the CAN bus. However, previous researches for WCRT analysis of CAN messages either assumed a priority queue or did not consider the offset. For this reason, in this paper we propose a WCRT analysis method for CAN messages with assigned offset in the FIFO queue. We first present a critical instant theorem, then we propose two algorithms for WCRT calculation based on the given theorem. Experimental results on generated message sets and a real message set have validated the effectiveness of the proposed algorithms.

Keywords: CAN, WCRT analysis, offset, FIFO queue

1. Introduction

In recent years, the electronic control systems of automobiles have become much more sophisticated and complex. A modern automobile may have as many as 70 electronic control units (ECUs) for various subsystems. Although some of these form independent subsystems, others are connected by the in-vehicle network, and communications among them are essential. The controller area network (CAN) is a widely used in-vehicle communication bus in modern automobiles.

Through an automobile communications network, many integrated services such as electronic stability control, pre-crash safety, etc., can be achieved by cooperation of the ECUs. However, number of messages and bus load of network has increased drastically. Because of real-time and safety requirements of automobiles, scheduling methods that make sure all messages could meet their deadlines are particularly important According to scheduling theory, a message set is schedulable only when the worst case response time (WCRT) of any message is below or equal to its deadline. For this purpose, WCRT analysis for CAN messages plays a crucial role in the design of electronic control systems of automobiles.

In previous work, most of the studies focus on the WCRT analysis based on the system using a priority queue. However, many commercial micro controller units (MCU) equipped with CAN controller also provide the first-in first-out (FIFO) queue to transmit message (e.g., the M16C/50 series MCU from Renesas [2]). Because the FIFO queue has faster queue management, using FIFO queue in CAN system can seem an attractive solution to improve the performance of the system [3]. To the best of our knowledge, only one published research discussed the WCRT analysis method of FIFO queued CAN message [4]. However, the study of Ref. [4] did not consider the messages with assigned offsets, so that the method can not analyze messages with offsets in a FIFO queue. For this reason, we propose the WCRT analysis method for the FIFO-based offset assigned CAN messages.

In this paper, we first present a critical instant theorem to locate the worst case of a given message. Then, based on the theorem we propose two algorithms for calculating the WCRT of messages in FIFO-based offset assigned CAN systems. Specifically, we propose an exact algorithm for accurate calculation of the WCRT, and an approximate algorithm for rapid estimation of the WCRT. The exact algorithm demands high computational complexity, which is suitable for evaluation of the approximate algorithm or for calculations in small systems. In contrast, the approximate could estimate the WCRT with limited errors and low computational complexity. And it is expected to be useful in large systems. We prove that the exact algorithm is accurate and the approximate algorithm is sufficiently safe for the WCRT analysis. Furthermore, we conducted experiments to validate the above two algorithms by using generated message sets and a real message set from an automaker.

The rest of this paper is organized as follows. Section 2 describes the terminology, notation and system models used in this paper. Section 3 includes a brief review of related work. Section 4 gives the critical instant theorem. Section 5 presents the

1 Department of Information Engineering, Graduate School of Information Science, Nagoya University, Nagoya, Aichi 464–8603, Japan
2 Center for Embedded Computing Systems, Graduate School of Information Science, Nagoya University, Nagoya, Aichi 464–8601, Japan
3 Graduate School of Engineering, Nagoya University, Nagoya, Aichi 464–8603, Japan
a) chenyang@ertl.jp

algorithms for the WCRT calculation. Section 6 shows the experiments and results. The extension with consideration of jitter is explained in Section 7, followed by a summary in Section 8. An appendix is attached at the end of this paper.

2. System Model, Terminology and Notation

2.1 The Controller Area Network

The Controller Area Network (CAN) is a serial broadcast bus that sends and receives short real-time control messages [1]. The CAN bus is designed to connect several stations and to operate at a maximum speed of 1 Mbit/sec. A simple CAN bus architecture is shown in **Fig. 1**.

Each CAN message is required to have a unique identifier (ID), which is either 11 bits (standard format) or 29 bits (extended format). When a CAN controller attempts to transmit a message, it has to wait until an idle bus period is detected. However, if two or more stations start to transmit at the same time, arbitration is triggered and the ID will be used as a priority flag to determine which message will be transmitted first among those contending for the bus. A message with a smaller ID will have higher priority as the CAN protocol.

Messages are queued in the stations before being transmitted to the CAN bus. The queue is memory implemented as dual ports and shared between the host processor and the CAN controller. An example is given in **Fig. 2**. In this example, the host processor queues the ID:1 message to the queue, where ID:2 and ID:3 messages already exist. As can be seen, after queuing the ID:1 message, while the CAN controller using the priority queue will attempt to transmit the highest-priority message (ID:1) to the CAN bus first, the CAN controller using the FIFO queue will attempt to transmit the first queued message (ID:2) to the CAN bus first.

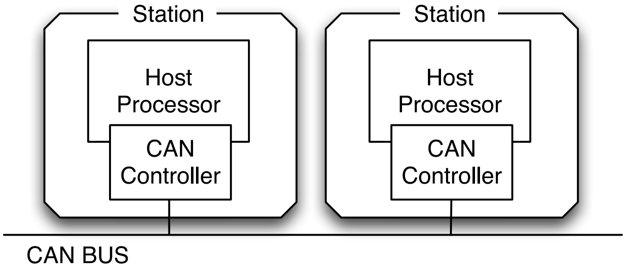
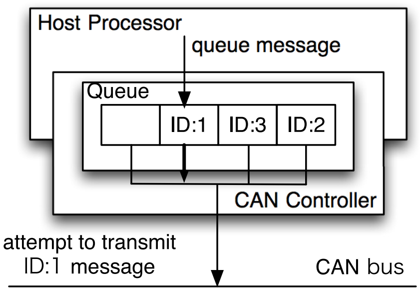
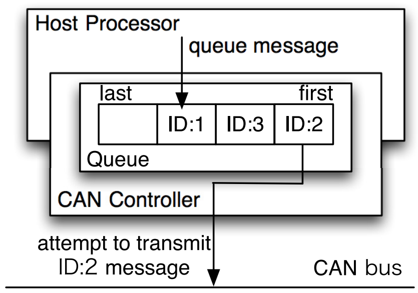


Fig. 1 CAN bus architecture.



(a) Priority queue



(b) FIFO queue

Fig. 2 Message queuing.

2.2 Basic Definitions in the CAN System

In this paper, the system for analysis is denoted by Θ that consists of a CAN bus and multiple CAN stations. Each station of Θ is denoted as U_I, where $I \in \mathcal{Z}^+$. $\mathcal{Z}$ denotes the set of all integers and $\mathcal{Z}^+$ is the set of all positive integers (i.e., 1, 2, 3...). Because message transmission is synchronized in same station but asynchronous in different stations, we assume that the phase ϕ_I ($\phi_I \geq 0$) occurs between the start time of the network and that of the station U_I. Since each station can be started at any instant after the network is initialized, U_I can have any ϕ_I. For convenience of analysis, we assume that the network has a global system clock and time 0 is defined as the instant when the CAN bus is ready to transmit messages.

For each U_I, it is assumed that at least one message is transmitted. A message is denoted as τ_i, where $i \in \mathcal{Z}^+$. The properties of τ_i consist of fixed priority P_i, transmission time on the CAN bus C_i [*1], period T_i and offset O_i. The value of P_i is equal to the ID of τ_i, and τ_i has a higher priority than τ_j if $P_i < P_j$. Since each CAN message has a unique ID in the network, for two messages τ_i, τ_j ($i \neq j$), $P_i \neq P_j$ holds.

In this paper, all messages are defined as periodic messages and reoccur infinitely. A frame is defined as each occurrence of a message. The frames of τ_i are $\tau_{i,0}, \tau_{i,1}, \ldots$ Define arrival and start are the instants of network time when a frame is requested to transmit, and starts to be transmitted from the queue to the CAN bus, respectively. Arrival and start of $\tau_{i,m}$ are denoted as $a_{i,m}$ and $s_{i,m}$. O_i thus is the interval between ϕ_I and $a_{i,0}$ (i.e., the arrival of the first frame of τ_i). T_i is the interval between $a_{i,m}$ and $a_{i,m+1}$. **Figure 3** is an example showing each property of a message τ_i.

The WCRT of any frame $\tau_{i,m}$ is the worst-case delay that $\tau_{i,m}$ may experience between arrival and complete transmission. Denote the WCRT of $\tau_{i,m}$ as $R_{i,m}$. The maximum $R_{i,m}$ is thus the WCRT of τ_i, denoted as R_i. Each τ_i is assigned a deadline D_i, which is the longest delay allowed for each frame $\tau_{i,m}$. We assume that $D_i \leq T_i$ holds for each message τ_i. Message τ_i is schedulable if $R_i \leq D_i$.

2.3 Assumptions

To simplify the analysis, it is assumed that there is no jitter on the arrivals of the messages (The extension with consideration of jitter is explained in Section 7). For the same purpose, all stations of the system are assumed to use FIFO to queue frames. Note that a system consisting of stations with FIFO or priority queue can be analyzed by extending our proposed method.

Also, it is assumed that in the FIFO queue the earlier a frame arrives, the earlier the frame can attempt to be transmitted to the

[*1] C_i can be calculated by considering the number of data bytes and bit-stuffing [8].

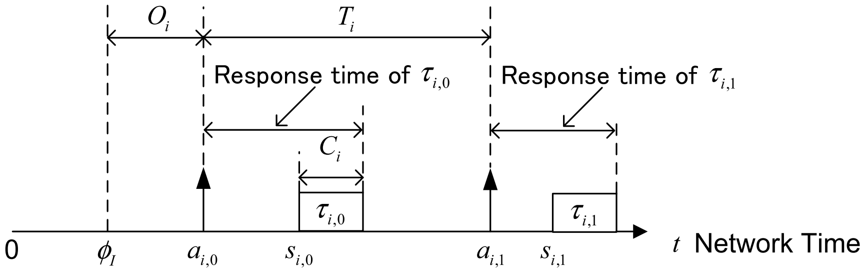


Fig. 3 Properties of message τ_i.

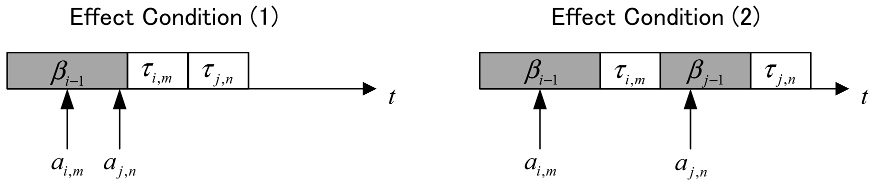


Fig. 4 Effects of the earlier queued frame.

CAN bus. However, if messages of the same station arrive at the same instant, it is assumed that the higher-priority message is queued earlier.

3. Related Work

Tindell et al showed how research about fixed priority scheduling for a single processor system could be adapted and applied to the scheduling of messages in CAN [5], [6]. They provided a method for calculating the WCRT of each message on the network. This research had significant flaws, which were found and corrected by Davis et al. in Refs. [7] and [8].

According to these studies, it was known that the message with low priority may be delayed by the high priority one, and this will become worse with the increase of bus load. To relieve this problem, (i.e., reduce the WCRT of low priority message), assigning offset to CAN messages has been proposed in Ref. [9]. Szakaly, Iiyama, as well as Du et al. independently provided WCRT calculation methods based on the priority queue assumption [10], [11], [13]. Their results showed that assigning offset to messages can reduce WCRT by 40%.

Note that all above studies assumed a priority queue for message transmission. Because the FIFO queue may introduce significant priority inversion comparing with the priority queue, these previous methods cannot work for CAN messages using the FIFO queue. To solve this problem, Davis et al proposed a WCRT analysis method for CAN message using the FIFO queue [4]. However, because the study of Ref. [4] did not consider messages with assigned offset, their method cannot analyze messages with offset. For this reason, this paper proposes WCRT analysis and calculation methods for messages with assigned offsets in a FIFO queue.

4. Proposed WCRT Analysis Method

According to the previous research, calculating the WCRT for a message can be achieved by the following two steps:

(1) Locate the worst case of the message.

(2) Calculate the response time of the message in its worst case.

In Refs. [7] and [8], the term of critical instant was employed to locate the worst case. It is defined as the instant at which a message arrival will have the largest response time. However, the FIFO-based and offset assigned system is too complex to be analyzed by the former critical instant definition. To solve this problem, we give a redefined critical instant first. Then we present a critical instant theorem to locate the worst case for the FIFO queued message.

4.1 The Definitions of Critical Instant and Critical Instant Candidate

For the purpose of analysis, we define the following terms:

Effect *Effect describes the delay of a frame, which is related with the earlier queued frame in the same station.*

Assume two frames $\tau_{i,m}, \tau_{j,n} \in U_I$, and $\tau_{i,m}$ is queued earlier than $\tau_{j,n}$. Then, $\tau_{i,m}$ affects $\tau_{j,n}$ if one of the following conditions holds:

(1) The instant, at which $\tau_{i,m}$ is completely transmitted, is later than $a_{j,n}$.

(2) A busy period (see A.1 of appendix) β_{j-1} continually occupies the CAN bus after $\tau_{i,m}$ is completely transmitted, and the finish of β_{j-1} is later than $a_{j,n}$.

The above two conditions are illustrated in **Fig. 4**.

Successive Frame Sequence *Assume two frames $\tau_{i,m}, \tau_{j,n} \in U_I$, and $\tau_{i,m}$ is queued earlier than $\tau_{j,n}$. If no other frame in U_I is queued between $\tau_{i,m}$ and $\tau_{j,n}$, then $\tau_{j,n}$ is the successive frame of $\tau_{i,m}$.*

Because queuing order is important for analysis, we denote a successive frame sequence as $\Gamma_0, \Gamma_1, ..., \Gamma_n$, which are sorted by the queued order.

Successive Affect Frame Sequence *In successive frames Γ_i, $\Gamma_{i+1}, ..., \Gamma_{i+m}$, if Γ_i affects Γ_{i+1}, Γ_{i+1} affects Γ_{i+2}, ..., Γ_{i+m-1} affects Γ_{i+m}, then $\Gamma_i, \Gamma_{i+1}, ..., \Gamma_{i+m}$ is the Γ_i successive affect frame sequence.*

The Γ_i successive affect frame sequence is denoted as Γ_{Ai-seq}. Any frame Γ_{i+m} in Γ_{Ai-seq} is denoted by $\Gamma_{A(i,m)}$. Also, $P_{A(i,m)}$, $C_{A(i,m)}$, $a_{A(i,m)}$, $s_{A(i,m)}$ and $R_{A(i,m)}$ denote the priority, transmission time on the CAN bus, arrival, start, and worst-case response time of $\Gamma_{A(i,m)}$, respectively. In addition, the level $P_{A(i,m)}$ busy period

Table 1 A message set example.

U_I	τ_i	P_i	T_i	C_i	O_i
U_1	τ_8	P_8	8	1	0
U_2	τ_5	P_5	8	1	0
U_2	τ_6	P_6	8	1	4
U_2	τ_2	P_2	8	1	5
U_3	τ_4	P_4	8	1	0
U_3	τ_1	P_1	8	1	4
U_4	τ_7	P_7	8	1	0
U_4	τ_3	P_3	8	1	1

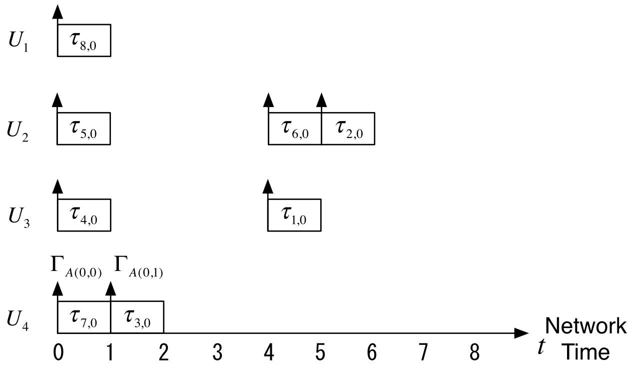


Fig. 5 An example of a critical instant candidate.

is denoted as $\beta_{A(i,m)}$, thus frame $\Gamma_{A(i,m)}$ that arrives in $\beta_{A(i,m)-1}$ can only access the CAN bus after the finish of $\beta_{A(i,m)-1}$.

Critical Instant *The critical instant of* $\Gamma_{A(i,m)}$ *is defined as the instant at which the arrival of* $\Gamma_{A(i,0)}$ *will lead to the largest response time of* $\Gamma_{A(i,m)}$.

Note that this definition differs from the previous one in two ways. First, the critical instant is for a frame but not a message. Second, the frame is involved in a successive affect frame sequence. However, in the offset assigned system, it is difficult to find the critical instant of $\Gamma_{A(i,m)}$ directly. Therefore, to locate the critical instant of $\Gamma_{A(i,m)}$, we give the following definition.

Critical Instant Candidates (CICs) of $\Gamma_{A(i,m)}$ *Assume* $\Gamma_{A(i,l_0)}$ *is the lowest-priority frame queued between* $\Gamma_{A(i,0)}$ *and* $\Gamma_{A(i,m)}$. *The CICs of* $\Gamma_{A(i,m)}$ *are defined as the instants that match the following conditions:*

(1) $\Gamma_{A(i,0)}$, *which is the first frame in* $\Gamma_{Ai\text{-}seq}$, *arrives simultaneously with any one of the frames belonging to the other stations with a priority higher than* $\Gamma_{A(i,l_0)}$.

(2) A frame with a priority lower than $\Gamma_{A(i,0)}$, *belonging to the other stations and having the largest transmission time, occupies the CAN bus just before the arrival of* $\Gamma_{A(i,0)}$.

We give an example of the *CIC*s. Assume a system consists of 4 stations and 8 messages, whose information is shown in **Table 1**. Focus on the WCRT analysis of the frame $\tau_{3,0}$ in U_4. Since $\tau_{7,0}$ is first queued before $\tau_{3,0}$ in the U_4, $\tau_{7,0}$ and $\tau_{3,0}$ are the successive affect frame sequence, which can be represented by $\Gamma_{A(0,0)}$ and $\Gamma_{A(0,1)}$, respectively. In particular, the $\tau_{7,0}$ can also be represented by $\Gamma_{A(i,l_0)}$ because it is the lowest-priority frame. From the definition above it is clear that $\tau_{3,0}$ has 6 *CIC*s which meet the following conditions on arrival time: $(a_{5,0} = a_{4,0} = a_{7,0})$, $(a_{6,0} = a_{4,0} = a_{7,0})$, $(a_{2,0} = a_{4,0} = a_{7,0})$, $(a_{5,0} = a_{1,0} = a_{7,0})$, $(a_{6,0} = a_{1,0} = a_{7,0})$, $(a_{2,0} = a_{1,0} = a_{7,0})$, respectively. As a concrete example, one of the *CIC*s is depicted in **Fig. 5**. The example corresponds to the above condition $(a_{5,0} = a_{4,0} = a_{7,0})$, which means that $\tau_{5,0}$, $\tau_{4,0}$, $\tau_{7,0}$ arrive simultaneously at network time 0. Note that $\tau_{8,0}$ is the frame that meets the second condition of *CIC*s definition, thus it is assumed to occupy the CAN bus just before the network time 0.

4.2 The Critical Instant Theorem

It is important to guarantee that the maximum response time of $\Gamma_{A(i,m)}$ always exists in its *CIC*s. To this end, we give the critical instant theorem and prove it as follows.

Theorem 1. *The critical instant of* $\Gamma_{A(i,m)}$ *occurs at one of the CICs of* $\Gamma_{A(i,m)}$.

Proof. The proof is achieved by following steps:

(1) Assume S_1 is any situation. S_2 is a variation of S_1, in which frames of U_I match the *CIC*s condition. S_3 is a variation of S_2, in which frames of U_J ($J \neq I$) match the *CIC*s condition also. S_4 is a variation of S_3, in which frames of all the stations match the *CIC*s condition.

(2) We prove that S_2 can always be found, in which response time of $\Gamma_{A(i,m)}$ is larger than or equal to it in S_1. Then, S_3 can always be found, in which response time of $\Gamma_{A(i,m)}$ is larger than or equal to it in S_2. Finally, S_4 can always be found too, in which response time of $\Gamma_{A(i,m)}$ is larger than or equal to it in S_3.

(3) Because all S_4 are included in the *CIC*s of $\Gamma_{A(i,m)}$ and response time of $\Gamma_{A(i,m)}$ is large than or equal to it in S_1, the critical instant of $\Gamma_{A(i,m)}$ occurs at one of the *CIC*s of $\Gamma_{A(i,m)}$.

In detail, we assume that frames of $\Gamma_{Ai\text{-}seq}$ belong to U_I, and there is any situation S_1, in which $\Gamma_{A(i,0)}$ arrives at t_1, and $\Gamma_{A(i,m)}$ finishes at t_e as shown in **Fig. 6**. Also, assume that t_0 is the last instant before $a_{A(i,0)}$ at which no frame with a priority higher than $\Gamma_{A(i,l_0)}$ is transmitted on the CAN bus. Meanwhile, $\Gamma_{A(i,l_0)}$ is the lowest-priority frame between $\Gamma_{A(i,0)}$ and $\Gamma_{A(i,m)}$. Because no frame is transmitted at network time 0, network time 0 meets the assumption of t_0. In other words, t_0 always exists.

Because the interval, from t_0 to the instant when $\Gamma_{A(i,l_0)}$ starts to be transmitted, is occupied by frames with priorities higher than $\Gamma_{A(i,l_0)}$, this interval is busy period $\beta_{A(i,l_0)-1}$. Again, search the lowest-priority frame $\Gamma_{A(i,l_1)}$, which is queued between $\Gamma_{A(i,l_0)}$ and $\Gamma_{A(i,m)}$. Then, the interval, from the instant when $\Gamma_{A(i,l_0)}$ is completely transmitted to the instant when $\Gamma_{A(i,l_1)}$ starts to be transmitted, is the busy period $\beta_{A(i,l_1)-1}$. Continue this searching until $\Gamma_{A(i,m)}$ becomes the lowest-priority frame after $\Gamma_{A(i,l_{n-1})}$. Let $\Gamma_{A(i,l_n)}$ be equal to $\Gamma_{A(i,m)}$, the interval $[t_0, t_e]$ can be described by $\beta_{A(i,l_0)-1}, ..., \beta_{A(i,l_n)-1}$, and $\Gamma_{A(i,l_0)}, ..., \Gamma_{A(i,l_n)}$, as shown in Fig. 6.

Assume that U_I starts earlier in situation S_2, so that $\Gamma_{A(i,0)}$ arrives at t_0, as shown in **Fig. 7**. The arrivals of frames in other stations do not change. Comparing with S_1, the length of each busy period in $[t_0, t_e]$ does not change in S_2. Hence $\Gamma_{A(i,l_n)}$ still finishes transmission at t_e. However, since $t_0 \leq t_1$, the arrivals of frames $\Gamma_{A(i,0)}, ..., \Gamma_{A(i,m)}$ in S_2 are earlier than or equal to that in S_1. Thus, the response time of $\Gamma_{A(i,l_n)}$ in S_2 is longer than or equal to that in S_1.

Assume that t_2 is an instant at which a frame of U_J ($J \neq I$) first arrives after t_0 in S_2 as shown in the S_2 of **Fig. 8**. Also assume that the frame of U_J first arrives at t_0 in situation S_3, as shown in

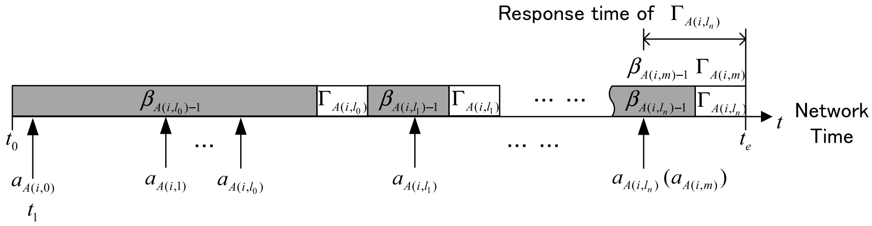


Fig. 6 Situation S_1.

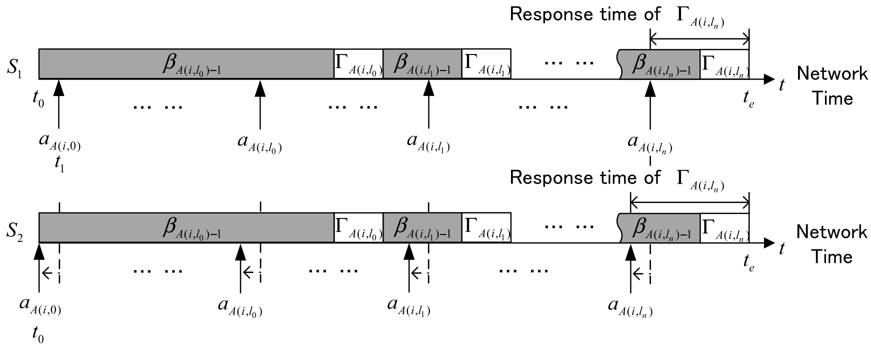


Fig. 7 From situation S_1 to S_2.

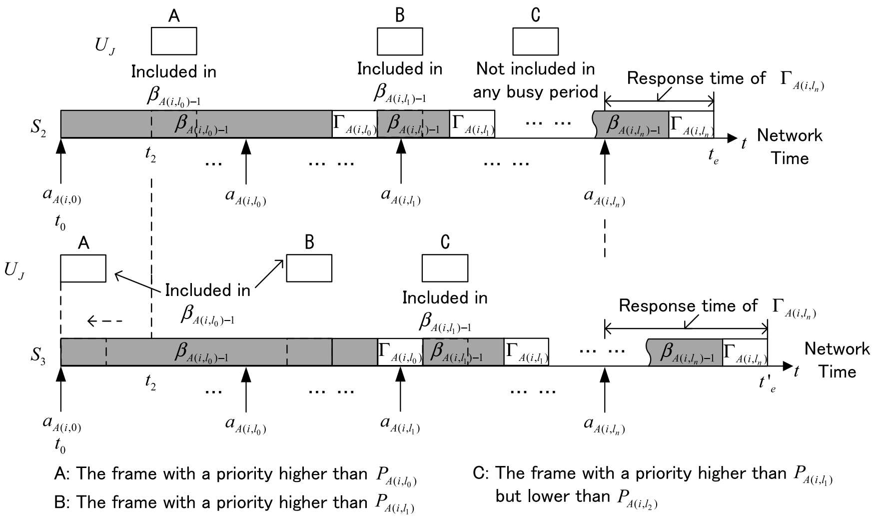


Fig. 8 From situation S_2 to S_3.

the S_3 of Fig. 8. The following changes occur from S_2 to S_3:

(1) Since $P_{A(i,l_0)} \geq P_{A(i,l_1)} \ldots \geq P_{A(i,l_n)}$, for each $\beta_{A(i,l_x)-1}$, the frames included in $\beta_{A(i,l_x)-1}$ in S_2 still exist in $\beta_{A(i,l_x)-1}$ or $\beta_{A(i,l_{x-1})-1}$ in S_3. Thus, no matter what the situation is, the sum of the lengths of all $\beta_{A(i,l_x)-1}$ does not change with these included frames. $\Gamma_{A(i,m)}$ (i.e., $\Gamma_{A(i,l_n)}$) will still finish transmission at t_e.

(2) The frames of U_J, which arrive after $\Gamma_{A(i,l_{x-1})}$ with a priority lower than $P_{A(i,l_x)}$ but higher than $P_{A(i,l_{x-1})}$, do not exist in any busy period in S_2, but may exist in $\beta_{A(i,l_{x-1})-1}$ in S_3. For example, as shown in Fig. 8, the frame of U_J, which arrive after $\Gamma_{A(i,l_1)}$ with a priority lower than $\Gamma_{A(i,l_2)}$ but higher than $\Gamma_{A(i,l_1)}$, does not exist in any busy period in S_2, but exist in the $\beta_{A(i,l_1)-1}$ in S_3. For this reason, the sum of the lengths of all $\beta_{A(i,l_x)-1}$ in S_3 may be larger than that in S_2. $\Gamma_{A(i,m)}$ will be transmitted completely at t'_e in S_3, which is later than t_e in S_2.

Finally, in situation S_4, the first frame in $[t_0, t'_e]$ of every other U_K arrives at t_0. And, a frame of other stations with a priority lower than $P_{A(i,0)}$ and the largest transmission time occupies the CAN bus just before t_0. Then, S_4 is a *CIC* of $\Gamma_{A(i,m)}$. The response time of $\Gamma_{A(i,m)}$ in S_4 is larger than or equal to that in S_1.

According to the above results, it is known that for any situation S_1, we can always find a relative situation S_4, in which the response time of $\Gamma_{A(i,m)}$ is larger than or equal to that in S_1. Because all the S_4 are included in the *CIC*s of $\Gamma_{A(i,m)}$, the WCRT of $\Gamma_{A(i,m)}$ always exists in its *CIC*s. □

5. Proposed Algorithms for WCRT Calculation

According to Theorem 1, the WCRT of messages in the station U_I can be calculated by the following steps:

(1) Define LCM_I as the least common multiple of periods of all messages in U_I. For each Γ_i ($\Gamma_i \in U_I$, Γ_i arrives between ϕ_I and $\phi_I + LCM_I$), focus on its successive affect frame se-

quence $\Gamma_{Ai\text{-}seq}$.

(2) For each $\Gamma_{A(i,m)}$ ($m = 0, 1, ...$) of each $\Gamma_{Ai\text{-}seq}$, locate all CICs of $\Gamma_{A(i,m)}$.

(3) Calculates $R_{A(i,m)}$ from these CICs.

(4) Compare $R_{A(i,m)}$ with WCRT of the message which $\Gamma_{A(i,m)}$ belongs to. Update WCRT of this message if $R_{A(i,m)}$ is larger.

In this method, the step 3 that calculates the WCRT for a given frame is the most important. For this calculation, we propose an exact algorithm and an approximate algorithm as follows.

5.1 Exact Algorithm

To obtain the exact WCRT of a given frame, one method is to calculate the $R_{A(i,m)}$ by checking all the CICs of $\Gamma_{A(i,m)}$ as the given theorem. The key to achieve this goal is to determine the latest finish of $\beta_{A(i,m)-1}$, at which the $\Gamma_{A(i,m)}$ will be able to transmit on the CAN bus. In a simple situation, $\Gamma_{A(i,m)}$ is the lowest-priority frame between $\Gamma_{A(i,0)}$ and $\Gamma_{A(i,m)}$, so that only one busy period $\beta_{A(i,m)-1}$ needs to be considered. However, as shown in Fig. 6, in general case multiple busy periods should be considered. In addition, the start of each $\beta_{A(i,l_x)-1}$ is related to the finish of $\beta_{A(i,l_{x-1})-1}$. Thus, to find the latest finish of $\beta_{A(i,m)-1}$, it is necessary to calculate the maximum sum of the lengths of all busy periods between $\Gamma_{A(i,0)}$ and $\Gamma_{A(i,m)}$.

In Refs. [11] and [12], the Interference Function (IF) and saturation addition were employed to calculate the length of a single busy period. IF is a function that represents the time in an interval when a set of frames interferes with the lower priority frame (see A.2 and A.4 of Appendix for details of IF and saturation addition). Because the original IF can not calculate the length of multiple busy periods, we extend the definition of IF to include more than one condition. The extended IF is denoted as $I_J^{ST}(t)\{(t_n^s, t_n^e, P_x^r)\}$, where ST is the network time at which the extended IF starts, $\{(t_n^s, t_n^e, P_n^r)\}$ is a set of conditions consisting of (t_0^s, t_0^e, P_0^r), (t_1^s, t_1^e, P_1^r),...,(t_n^s, t_n^e, P_n^r). According to the conditions, $I_J^{ST}(t)\{(t_n^s, t_n^e, P_n^r)\}$ is created by considering the frames that arrive in $[t_x^s, t_x^e]$ with a priority higher than $P_x^r (0 \le x \le n)$. The t_x^s, t_x^e are relative time to the ST.

The exact algorithm using the extended IF is presented in **Fig. 9**. For each CIC of $\Gamma_{A(i,m)}$, initialize the start frame Γ_s, the end frame Γ_e and parameter x to $\Gamma_{A(i,0)}$, $\Gamma_{A(i,m)}$ and 0, respectively (lines 02, 03). In line 05, search the lowest-priority frame $\Gamma_{A(i,l_x)}$ between Γ_s and Γ_e. Then, initialize the IF conditions t_x^s, P_x^r to the start of Γ_s and $P_{A(i,l_x)}$ (line 06). And initialize t_x^e to a sufficiently big value (i.e., $LCM_{\tau_i \in \Theta}\{T_i\}$), so that $\beta_{A(i,l_x)-1}$ ends before t_x^e (line 07).

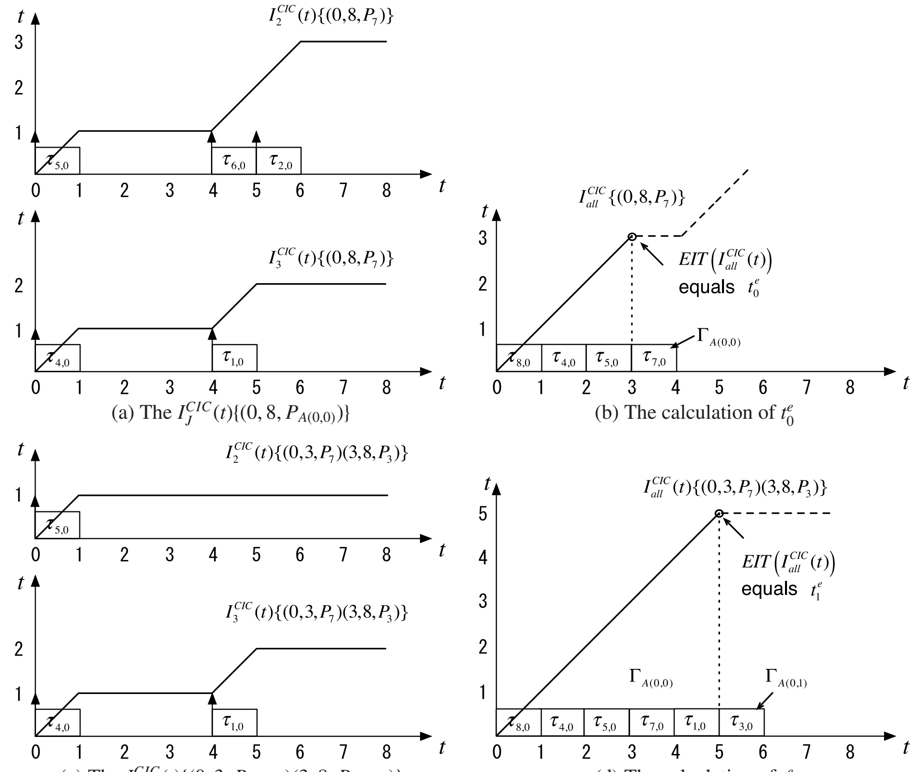
According to the definition of CIC, calculation of the finish of $\beta_{A(i,l_x)-1}$ relates with the following interference time: (1) delay caused by the frames of other stations with higher priority, which will be included in the $\beta_{A(i,l_0)-1}, ..., \beta_{A(i,l_x)-1}$ (line 08); (2) delay caused by the earlier queued frames of self station, which can be calculated by the sum of the transmission time of frames $\Gamma_{A(i,0)}, ..., \Gamma_{A(i,l_x-1)}$ (line 09); (3) delay caused by the frame of other stations with the largest transmission time and a priority lower than $\Gamma_{A(i,0)}$ (line10). The saturation addition of these elements is denoted as $I_{all}^{CIC}(t)$. Then, the finish of $\beta_{A(i,l_x)-1}$ can be calculated by $EIT(I_{all}^{CIC}(t))$. $EIT(I_{all}^{CIC}(t))$ is the operation that finds the first instant at which the slope of the function $I_{all}^{CIC}(t)$ becomes 0. Meanwhile, its result is the earliest idle time of CAN bus, at which $\Gamma_{A(i,l_x)}$ can be transmitted (see A.4 of Appendix for details of EIT). Because frames that arrive after the $EIT(I_{all}^{CIC}(t))$ cannot interfere with $\Gamma_{A(i,l_x)}$, t_x^e is updated to $EIT(I_{all}^{CIC}(t))$ (line 12). Then, Γ_s is updated to $\Gamma_{A(i,l_x)}$, and the parameter x is increased 1 (line 13). The algorithm continues to calculate a new finish of $\beta_{A(i,l_x)-1}$ with the updated IF conditions until Γ_s equals Γ_e. After the while loop, the t_x^e will be the finish of $\beta_{A(i,m)-1}$. Then the start time of $\Gamma_{A(i,m)}$ can be calculated by addition of t_x^e and $s_{A(i,0)}$ (line 15), because the t_x^e is a relative time to the CIC (i.e., $s_{A(i,0)}$). Finally, the response time of $\Gamma_{A(i,m)}$ of the current CIC is calculated and updated to $R_{A(i,m)}$ in lines 16 and 17. When all candidates have been checked, the maximum response time in all CICs will be the WCRT of the $\Gamma_{A(i,m)}$.

01 $R_{A(i,m)} \leftarrow 0$.
02 **for all** CICs of $\Gamma_{A(i,m)}$ **do**
03 $\quad\Gamma_s \leftarrow \Gamma_{A(i,0)}$; $\Gamma_e \leftarrow \Gamma_{A(i,m)}$; $x \leftarrow 0$
04 $\quad$**while** $\Gamma_s \neq \Gamma_e$ **do**
05 $\quad\quad$Searching $\Gamma_{A(i,l_x)}$ which is the lowest-priority frame queued between Γ_s and Γ_e
06 $\quad\quad t_x^s \leftarrow$ start of Γ_s; $P_x^r \leftarrow P_{A(i,l_x)}$
07 $\quad\quad t_x^e \leftarrow LCM_{\tau_i \in \Theta}\{T_i\}$
08 $\quad\quad$Create $I_J^{CIC}(t)\{(t_0^s, t_0^e, P_0^r), ..., (t_x^s, t_x^e, P_x^r)\}$ for each $U_J (J \neq I)$, calculate saturation sum of $I_J(t)$ of all U_J
09 $\quad\quad$Calculate sum of $C_{A(i,0)}, C_{A(i,1)}, ..., C_{A(i,l_x-1)}$
10 $\quad\quad$Search the max $C_k (P_k > P_{A(i,0)})$ in other stations
11 $\quad\quad I_{all}^{CIC}(t) \leftarrow$ Saturation sum results of lines 08-10
12 $\quad\quad t_x^e \leftarrow EIT(I_{all}^{CIC}(t))$
13 $\quad\quad\Gamma_s \leftarrow \Gamma_{A(i,l_x)}$; $x \leftarrow x + 1$
14 $\quad$**end while**
15 $\quad s_{A(i,m)} \leftarrow t_x^e + s_{A(i,0)}$
16 $\quad$**if** $(s_{A(i,l_m)} + C_{A(i,m)} - a_{A(i,m)} > R_{A(i,m)})$ **then**
17 $\quad\quad R_{A(i,m)} \leftarrow s_{A(i,l_m)} + C_{A(i,m)} - a_{A(i,m)}$
18 $\quad$**end if**
19 **end for**
20 **return** $R_{A(i,m)}$

Fig. 9 The exact algorithm.

Consider an example to calculate the response time of $\tau_{3,0}$ in the CIC of Fig. 5. Because $\tau_{3,0}$ is involved in a successive affect frame sequence: $\Gamma_{A(0,0)}$ ($\tau_{7,0}$) and $\Gamma_{A(0,1)}$ ($\tau_{3,0}$), 2 busy periods $\beta_{A(0,0)-1}, \beta_{A(0,1)-1}$ should be considered. First, to calculate the finish of $\beta_{A(0,0)-1}$, the IF condition is initialized to $\{(0, 8, P_7)\}$, since the start of CIC is 0 and the $LCM_{\tau_i \in \Theta}\{T_i\}$ is 8. Then, the IF of each station is created as shown in **Fig. 10** (a). Next, the $I_{all}^{CIC}\{(0, 8, P_7)\}(t)$ can be calculated by saturation addition of all $I_J^{CIC}\{(0, 8, P_7)\}$ and C_8, as shown in Fig. 10 (b). Because $EIT(I_{all}^{CIC}(t))$ is the finish of frames that interfere with $\Gamma_{A(0,0)}$, network time 3 is the end of $\beta_{A(0,0)-1}$, and $\tau_{7,0}$ starts to transmit at this instant.

Since the end of $\beta_{A(0,0)}$ is known, the IF conditions are updated to $\{(0, 3, P_7)\ (3, 8, P_3)\}$. Then, to calculate the finish of $\beta_{A(0,1)-1}$, $I_2^{CIC}(t)\{(0, 3, P_7)(3, 8, P_3)\}$ and $I_3^{CIC}(t)\{(0, 3, P_7)(3, 8, P_3)\}$ are created as shown in Fig. 10 (c). Next, $I_{all}^{CIC}\{(0, 3, P_7)(3, 8, P_3)\}$ is calculated by saturation addition of $I_2^{CIC}(t)\{(0, 3, P_7)(3, 8, P_3)\}$,



(a) The $I_J^{CIC}(t)\{(0, 8, P_{A(0,0)})\}$ (b) The calculation of t_0^e

(c) The $I_J^{CIC}(t)\{(0, 3, P_{A(0,0)})(3, 8, P_{A(0,1)})\}$ (d) The calculation of t_1^e

Fig. 10 An example of using IF to calculate response time of $\Gamma_{A(0,1)}$ in its CIC.

$I_3^{CIC}(t)\{(0, 3, P_7)(3, 8, P_3)\}$, C_7, and C_8, as shown in Fig. 10 (d). Thus, the end of $\beta_{A(0,1)-1}$ is given by the $EIT(I_{all}^{CIC}(t))$, and the response time of $\tau_{3,0}$ (i.e., $\Gamma_{A(0,1)}$) in this CIC is equal to 5, as shown in Fig. 10 (d).

In the same way, the response time of $\tau_{3,0}$ in its other 5 CICs can be calculated. Finally, the maximum response time in the 6 CICs will be the WCRT of $\tau_{3,0}$, which is 7 in this example.

5.2 Approximate Algorithm

As mentioned before, the exact algorithm has to check every CIC of $\Gamma_{A(i,m)}$. However, the number of CICs will become huge with the increase in the number of messages in a large system, which will result in unaffordable calculation time. Therefore, we propose an approximate algorithm to speed up the calculation by using the Maximum Interference Function (MIF) instead of the IF. MIF is a function that represents the maximum time in a interval when a set of frames interferes with the lower-priority frame (see A.3 and A.4 of Appendix for details of MIF). MIF based calculation needs only one operation no matter how many CICs exist. Because MIF is the max of IFs, MIF is also extended to have multiple conditions and denoted as $M_J(t)\{(t_n^s, t_n^e, P_n^r)\}$ as done for IF.

The MIF based algorithm for calculation of $R_{A(i,m)}$ is given in **Fig. 11**. In this algorithm, the first step initializes the start frame Γ_s, end frame Γ_e and MIF conditions (lines 01–05), which is the same as the exact algorithm. Second, the $M_J(t)\{(t_0^s, t_0^e, P_0^r), ..., (t_x^s, t_x^e, P_x^r)\}$ of each station U_J ($J \neq I$) is created by line 06–09. The finish of $\beta_{A(i,l_x)-1}$ is calculated by $EIT(M_{all}(t))$ (lines 10–13). Because the $EIT(M_{all}(t))$ is the latest finish time of $\beta_{A(i,l_x)-1}$. the frames that arrive after the

01 $\Gamma_s \leftarrow \Gamma_{A(i,0)}, \Gamma_e \leftarrow \Gamma_{A(i,m)}, x \leftarrow 0$
02 **while** $\Gamma_s \neq \Gamma_e$ **do**
03 Searching $\Gamma_{A(i,l_x)}$ which is the lowest-priority frame queued between Γ_s and Γ_e
04 $t_x^s \leftarrow$ start of Γ_s, $P_x^r \leftarrow P_{A(i,l_x)}$
05 $t_x^e \leftarrow LCM_{\tau_i \in \Theta}\{T_i\}$

06 **for** each $U_J (J \neq I)$ **do**
07 Create all $I_J^{ST}(t)\{(t_0^s, t_0^e, P_0^r), ..., (t_x^s, t_x^e, P_x^r)\}$, in which $ST = a_{j,n}$ ($a_{j,n}$ subject to $\phi_J \leq a_{j,n} < \phi_J + LCM_J,\ P_j < P_0^r,\ \tau_j \in U_J$)
08 Create $M_J(t)\{(t_0^s, t_0^e, P_0^r), ..., (t_x^s, t_x^e, P_x^r)\}$ as the max of all the $I_J^{ST}(t)\{(t_0^s, t_0^e, P_0^r), ..., (t_x^s, t_x^e, P_x^r)\}$ of line 07
09 **end for**
10 Calculate saturation sum of $M_J(t)$ of all U_J
11 Calculate sum of $C_{A(i,0)}, C_{A(i,1)}, ..., C_{A(i,l_x-1)}$
12 Calculate max $C_k(P_k \geq P_{A(i,0)}, \tau_k \in \Theta)$
13 $M_{all}(t) \leftarrow$ Saturation sum results of lines 10-12
14 $t_x^e \leftarrow EIT(M_{all}(t))$

15 $\Gamma_s \leftarrow \Gamma_{A(i,l_x)}, x \leftarrow x + 1$
16 **end while**
17 $s_{A(i,m)} \leftarrow t_x^e + a_{A(i,0)}$
18 $R_{A(i,m)} \leftarrow s_{A(i,m)} + C_{A(i,m)} - a_{A(i,m)}$
19 **return** $R_{A(i,m)}$

Fig. 11 The approximate algorithm.

$EIT(M_{all}(t))$ cannot interfere with $\Gamma_{A(i,l_x)}$. t_x^e is thus updated to $EIT(I_{all}^{CIC}(t))$ (line 14). For the next while loop, Γ_s is updated to $\Gamma_{A(i,l_x)}$, and the parameter x is increased 1 (line 15). The algorithm continues to calculate a new t_x^e until Γ_s is equal to Γ_e. Then, the final t_x^e will be the latest finish of $\beta_{A(i,m)-1}$. Finally, $R_{A(i,m)}$ can be calculated by using the above results as shown in lines 17 and 18.

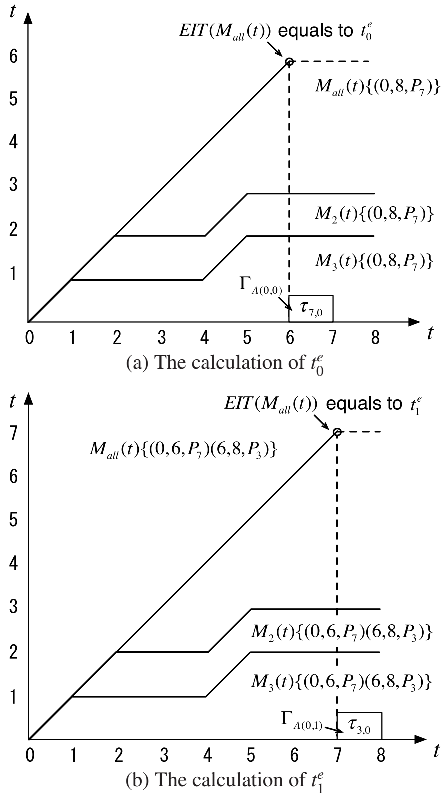


Fig. 12 An example of using MIF to calculate $R_{A(0,1)}$.

Consider an example using the message set in Table 1. We calculate the WCRT of $\tau_{3,0}$ in U_4. Because $\tau_{3,0}$ is involved in a successive affect frame sequence, 2 busy periods $\beta_{A(0,0)-1}$, $\beta_{A(0,1)-1}$ should be considered.

First, the algorithm creates the MIF for U_2 and U_3 with the condition $\{(0, 8, P_7)\}$. Note that MIF of U_1 is ignored here because it equals zero. Then, $M_{all}(t)\{0, 8, P_7\}$ is calculated by saturation addition of $M_2(t)\{(0, 8, P_7)\}$, $M_3(t)\{(0, 8, P_7)\}$, and C_8. Next, the latest finish of $\beta_{A(0,0)-1}$ is calculated by the $EIT(M_{all}(t)\{0, 8, P_7\})$ as shown in **Fig. 12** (a). In other words, at this instant, i.e., network time 6, the $\tau_{7,0}$ can be transmitted on the CAN bus. Then, conditions of MIF are updated to $\{(0, 6, P_7)(6, 8, P_3)\}$ to calculate the next busy period. Finally, the latest finish of $\beta_{A(0,1)-1}$ is calculated by $EIT(M_{all}(t)\{(0, 6, P_7)(6, 8, P_3)\}$ as shown in Fig. 12 (b). As can be seen, the $\tau_{3,0}$ can be transmitted at network time 7. Therefore, the WCRT of $\tau_{3,0}$ is 7.

From the above example, it is clear that while the IF based exact algorithm needs to check all 6 CICs to calculate the WCRT of $\tau_{3,0}$, the MIF based algorithm only requires one calculation of MIF for each station. Note that although the results of the approximate algorithm are not completely accurate, they are equal to or larger than the real WCRT in all cases according to the feature of MIF operation (see A.3 of Appendix for details). Therefore its results are sufficiently safe from the WCRT analysis point of view.

5.3 Computational Complexity Analysis

To analyze the computational complexity of the proposed two algorithms, let us assume that n stations exist in the network, $\Gamma_{A(i,m)}$ is a frame of U_I, which belongs to $\Gamma_{Ai\text{-}seq}$. Also, assume that each station U_J has N_J frames with priorities higher than $P_{A(i,l_0)}$. The $P_{A(i,l_0)}$ is the priority of $\Gamma_{A(i,l_0)}$ which is the lowest-priority frame between $\Gamma_{A(i,0)}$ and $\Gamma_{A(i,m)}$. The number of CICs of $\Gamma_{A(i,m)}$, which represents the computational complexity of the exact algorithm, is as follows:

$$Complexity_{exact} = \prod_{J\in\Theta, J\neq I} N_J \tag{1}$$

However, the computational complexity of the approximate algorithm is given as follows:

$$Complexity_{approximate} = \sum_{J\in\Theta, J\neq I} N_J \tag{2}$$

From the above equations, it is clear that the approximate algorithm can greatly decrease the computational complexity in a large system.

6. Evaluation

In order to validate the efficiency of the proposed methods, experiments are conducted by using message sets generated by NETCARBENCH [16] and a real message set provided by an automaker. All the experiments are performed on a computer with an Intel Core i5 2.67 GHz processor. As mentioned in Section 2.3, all stations are assumed to use the FIFO queue.

6.1 Experiment of NETCARBENCH-generated Message Sets

In experiment 1, WCRT of message is analyzed on 10 small message sets generated by NETCARBENCH. All the message sets are configured as a typical 500 kbps powertrain network with a bus load of 20–25%, station number 3–5 and message number 50–76. Deadlines of messages are assumed to be equal to their periods. Priorities of messages are assigned based on the deadline monotonic algorithm: the shorter deadline the message has, the higher priority the message is assigned. The offset of each message is assigned based on the method of Ref. [9]. Basically, the method tries to assign offsets in such a way that the arrivals of any two messages are as far as possible. We briefly summarize the method of Ref. [9] as follows:

(1) Initialize an empty offset assignment record for all messages.
(2) In each U_I, find the τ_i that is the shortest period message of U_I and has not been assigned offset.
(3) Based on the offset assignment record, find the longest interval (t_0, t_1) of $[0, T_i)$ in which no message arrives.
(4) Assign $(t_0 + t_1)/2$ to O_i and update the offset assignment record.
(5) Repeat step (2) to (4) until all messages have their offsets assigned.

Results of experiment 1 are shown in **Table 2**. While the exact algorithm takes an average time of 492.82 seconds to finish the calculation, the approximate algorithm only needs an average time of 1.02 seconds. As for accuracy, the approximate algorithm has an average of 7.66% messages with different results from the the exact algorithm. Specifically, although two message sets have no errors, the approximate algorithm achieves an average error of

Table 2 Experiment 1: Error and run time comparison of exact and approximate algorithm based on 10 message sets generated by NETCARBENCH.

Algorithm	Message with error	Max Error	Average Error	Run time
Exact	-	-	-	492.82 s
Approximate	7.66%	8.3%	1.95%	1.02 s

Table 3 Experiment 2: Error and run time comparison of exact and approximate algorithm based on a real message set.

Algorithm	Message with error	Max Error	Average Error	Run time
Exact	-	-	-	7 days
Approximate	0%	0%	0%	17.46 s

Table 4 Experiment 3: WCRT comparison of a real message set with and without offset.

	Message	Average WCRT	Maximum WCRT
Decrease rate	96.97%	42.56%	64.21%

1.95%, and a max error of 8.3% comparing with the exact algorithm. The error of a message τ_i is calculated by the following formula:

$$Error_i = (R_i^{approximate} - R_i^{exact})/R_i^{exact} * 100\% \tag{3}$$

where $R_i^{approximate}$ and R_i^{exact} are the WCRT of τ_i calculated by approximate algorithm and exact algorithm, respectively.

6.2 Experiment of Automaker-provided Message Set

To validate the efficiency of the proposed algorithms on a real system, we used the message set provided by an automaker, which is composed of 14 stations and 66 messages, and has 53.3% bus load. All message properties, including the offset, were configured by the automaker. Deadlines of messages are configured to equal to their periods. Priorities of messages are assigned mainly based on the deadline monotonic algorithm. Offset assignment of the messages are similar to the NETCARBENCH method.

Because the system is too large to check all stations by the exact algorithm, we tested one station of this network system, which includes 15 messages. As shown in **Table 3**, while the exact algorithm required 7 days to finish the analysis of WCRT for 15 messages, the approximate algorithm only required 17.46 seconds. Note that although there are no errors on the 15 messages, it does not mean that the approximate algorithm can always obtain the same results as the exact algorithm as indicated in the first experiment.

6.3 Experiment of Effectiveness of Offset Assignment

To confirm the effectiveness of assigning offset to messages in a FIFO queue system, we employed the same message set as used in the second experiment which is provided by an automaker. **Table 4** illustrates the decrease rate of average WCRT and maximum WCRT of the messages set after assigning offset. As can be seen, there are 96% messages' WCRT are decreased after being assigned offset. In these messages, average decrease rate of WCRT is 42%, maximum decrease rate of WCRT is 64%, The results confirmed that assigning offset can also greatly decrease WCRT of messages in the FIFO queue system.

7. Extension with Consideration of Jitter

In this paper, we assumed that there is no jitter on the arrivals of the messages for the purpose of simplifying the analysis. However, considering that the existence of jitter can affect WCRT of messages, we explain the extension with consideration of jitter in this section.

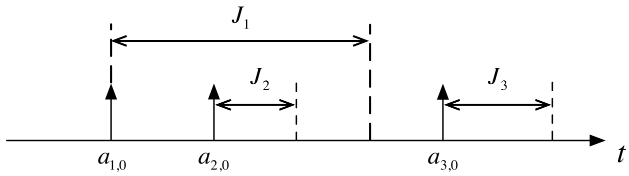


Fig. 13 An example showing effect of jitter on the queuing order of messages.

7.1 Effect of Jitter on the Proposed Method

For message τ_i, the queueing process takes a bounded amount of time, between 0 and J_i, before τ_i is queued available for transmission. J_i is referred to as the maximum queuing jitter of τ_i. Considering jitter, frame $\tau_{i,m}$ may arrives at any time between $[a_{i,m}, a_{i,m} + J_i]$.

The occurrence of jitter leads arrival times and queuing order of messages to become changeable, which betrays our assumptions. The proposed definitions, successive frame sequence and successive affect frame sequence, thus are not suitable for the jitter model. However, this problem can be solved by extending the proposed definitions.

7.2 Extended Successive Affect Frame Sequences

As we know, J_i of each τ_i is fixed when a message set is generated. For this reason, all the possible queuing orders of messages in each station are finite and can be analyzed. We denote successive frame sequences of messages in same station as $(\Gamma_0^k, ..., \Gamma_n^k)$ $(k = 0, 1, ...)$. Each sequence is referred as a queuing order of messages in this station. For example, assume $\tau_{1,0}$, $\tau_{2,0}$, $\tau_{3,0}$ are frames of same station, arrival time and maximum jitters of these messages are shown in **Fig. 13**. It is clear that there are 2 queuing orders of messages in this station: $(\tau_{1,0}(\Gamma_0^0), \tau_{2,0}(\Gamma_1^0), \tau_{3,0}(\Gamma_2^0))$ and $(\tau_{2,0}(\Gamma_0^1), \tau_{1,0}(\Gamma_1^1), \tau_{3,0}(\Gamma_2^1))$.

In each successive frame sequence $(\Gamma_0^k, ..., \Gamma_n^k)$, the successive affect frame sequence of Γ_i^k, denoted as Γ_{Ai-seq}^k, can be found by the definition of Effect in Section 4.1.

7.3 Worst-case Jitter Occurred *CIC*s of $\Gamma_{A(i,m)}^k$

Based on the extended successive affect frame sequences, *CIC*s of $\Gamma_{A(i,m)}^k$ can be found by considering all the queuing orders of messages in other stations. In each *CIC* of $\Gamma_{A(i,m)}^k$, because queuing order of messages is unique, WCRT of $\Gamma_{A(i,m)}^k$ will be largest if frames arriving at the CIC have the jitter as large as possible, and the frames arriving after the CIC have the jitter as small as possible*2. This kind of CICs is defined as the worst-case jitter occurred CICs of $\Gamma^k_{A(i,m)}$. Because queuing order and arrival times of messages are fixed in each worst-case jitter occurred CIC of $\Gamma^k_{A(i,m)}$, WCRT of $\Gamma^k_{A(i,m)}$ thus can be calculated by our proposed algorithms.

However, the arriving order of messages may have many situations if the maximum jitters of messages are large. In this case, finding the worst case arriving order firstly then calculating WCRT of $\Gamma^k_{A(i,m)}$ will be a solution for speeding up the calculation. This will be considered as future work.

8. Conclusion

In this paper, we proposed a WCRT analysis method for messages in the FIFO-based and offset assigned CAN systems. We first gave a critical instant theorem and proved it, then we proposed two algorithms for the WCRT calculation based on the given theorem. The exact algorithm can obtain accurate results with a large computational cost, which is suitable for a small system. In contrast, the approximate algorithm can analyze a larger system with limited errors and low computational complexity. Experimental results on generated message sets and a real message set have validated the effectiveness of the proposed two algorithms. Also, an experiment on a real message set showed assigning offset to the messages can decrease the WCRT significantly in the FIFO queued system.

In future work, we will focus on the error analysis for the approximate algorithm and look for ways to decrease the computational complexity of the exact algorithm. Also, fast calculation algorithms and evaluations of the jitter model are considered.

Reference

[1] International Organization for Standardization, Road vehicles: Controller area network (CAN), Part 1: Data link layer and physical signaling, ISO IS11898-1 (2003).

[2] M16C/50 Series microcontrollers of Renesas, Documents available from ⟨http://www.renesas.com/products/mpumcu/m16c/m16c50/Documentation.jsp⟩.

[3] Natale, M.D.: Understanding and using the Controller Area network, available from ⟨http://inst.eecs.berkeley.edu/~ee249/fa08/Lectures/handout_canbus2.pdf⟩.

[4] Davis, R.I., Kollmann, S., Pollex, V. and Slomka, F.: Controller Area Network (CAN) Schedulability Analysis with FIFO queues, University of York, Department of Computer Science Technical Report, YCS-2010-462 (Jan. 2011).

[5] Tindell, K., Hansson, H. and Wellings, A.J.: Analysing Real-time communications: Controller Area Network (CAN), *Proc. 15th IEEE Real-Time Systems Symposium* (*RTSS'94*), pp.259–263, IEEE Computer Society Press (1994).

[6] Tindell, K.W., Burns, A. and Wellings, A.J.: Calculating Controller Area Network (CAN) message response times (1995), *Control Engineering Practice*, Vol.3, No.8, pp.1163–1169 (Aug. 1995).

[7] Bril, R.J., Lukkien, J.J., Davis, R.I. and Burns, A.: Message response time analysis for ideal controller area network (CAN) refuted, *Proc. 5th International Workshop on Real Time Networks* (*RTN*) (2006).

[8] Davis, R.I., Burns, A., Bril, R.J. and Lukkien, J.J.: Controller area network (CAN) schedulability analysis: Refuted, revisited and revised, *Proc. Real-time Systems*, Vol.35, Issue 3, pp.239–272 (Apr. 2007), ISSN:0922-6443.

[9] Grenier, M., Havet, L. and Navet, N.: Scheduling frames with offsets in automotive systems: A major performance boost, *Automotive Embedded Systems Handbook*, Navet, N. and Simonot-Lion, F. (Eds.), CRC Press/Taylor and Francis (2008).

[10] Szakaly, A.: Response time analysis with offsets for CAN, Master Thesis of Department of Computer Engineering, Chalmers University of Technology, pp.1–67 (Nov. 2003).

[11] Iiyama, S., Tomiyama, H., Takada, H., Kido, M. and Hosotani, I.: Response time analysis for grouped CAN messages with offsets, *IPSJ Journal, Computing System*, Vol.45, pp.455–464 (2004) (in Japanese).

[12] Iiyama, S.: Applying Real-Time Scheduling Theories to Automotive Control Systems, PhD thesis of Department of Information Engineering, Toyohashi University of Technology, pp.40–76 (2004) (in Japanese).

[13] Du, L. and Xu, G.: Worst case response time analysis for CAN messages with offsets, *Vehicular Electronics and Safety* (*ICVES*), pp.41–45 (Nov. 2009).

[14] Harbour, M.G., Klein, M.H. and Lehoczky, J.P.: Fixed priority scheduling of periodic tasks with varying execution priority, *Proc. 12th IEEE Real-Time Systems Symposium*, pp.116–128, IEEE Computer Society Press (Dec. 1991).

[15] Takada, H. and Sakamura, K.: Schedulability of generalized multiframe task sets, under static priority assignment, *Proc. Real-time Computing Systems and Applications*, pp.80–86 (1997).

[16] Braun, C., Havet, L. and Navet, N.: NETCARBENCH: A benchmark for techniques and tools used in the design of automotive communication systems, *Proc. 7th IFAC International Conference on Fieldbuses and Networks in Industrial and Embedded Systems* (*FeT'07*) (Nov. 2007). Software and manual, available from ⟨http://www.loria.fr/navet/netcarbench/⟩.

*2 Similar conclusion about the worst-case jitter has been summarized by Refs. [5], [6], [8], [10].

Appendix

A.1 Busy Periods

In CAN system, a busy period is a period of time during which the CAN bus is continually occupied by frame transmission. An extension to this concept, the level i busy period, is defined as a period of time during which the CAN bus is completely occupied by the transmission of messages with priority P_i or higher [14]. The level i busy period is denoted by β_i. A frame $\tau_{i,m}$, which arrives during β_{i-1}, will be able to gain access to the CAN bus after the end of the β_{i-1}.

A.2 Interference Function

The Interference Function (IF) is defined as a function representing the time in an interval, in which a set of tasks interferes with the lower-priority task [15]. When it was employed in a CAN system, IF is redefined as a function that represents the time in an interval when a set of frames interferes with the lower-priority frame [12]. IF of frames in U_J is denoted as $I^{ST}_J(t)\{t^s, t^e, P_i\}$, where ST represents the network time of start point, and $\{t^s, t^e, P_i\}$ represents the condition of the IF. Thus, $I^{ST}_J(t)\{t^s, t^e, P_i\}$ denotes the IF of frames in U_J, which arrive in $[t^s, t^e]$ and interfere with the frames having a priority lower than P_i. Note that because the t^s and t^e are relative time to the ST, their network time are $ST + t^s$ and $ST + t^e$, respectively.

A.3 Maximum Interference Function

The Maximum Interference Function (MIF) is defined as a function representing the maximum time in an interval, in which a set of messages interferes with the lower-priority frame [15]. When employed in the CAN system, MIF is redefined as a function that represents the maximum time in an interval when a set of frames interferes with the lower-priority frame [11], [12]. Denote $M_J(t)(t^s, t^e, P_i)$ as the MIF of frames in U_J, which interfere with the frames having a priority lower than P_i. $M_j(t)(t^s, t^e, P_i)$ is calculated by following formula:

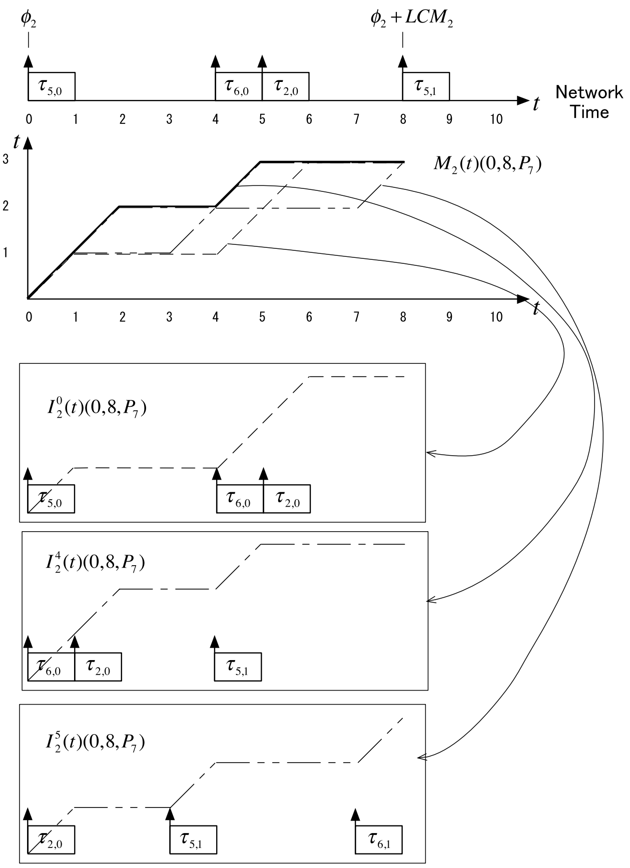


Fig. A·1 An example of the Maximum Interference Function.

$$M_J(t)(t^s, t^e, P_i) = \max_{ST=a_{j,n}, \phi_J \le a_{j,n} < \phi_J + LCM_J} \{I_J^{ST}(t)(t^s, t^e, P_i)\} \tag{A.1}$$

In the formula, $\{I_J^{ST}(t)\{(t^s, t^e, P_i\}\}$ is the set of IFs that start from each $a_{j,n}$. The $a_{j,n}$ is the arrival of frame $\tau_{j,n}$ in U_J that has a priority higher than P_i.

An example of IF and MIF is given in **Fig. A·1** by using the message set of Table 1. This example calculates the maximum time in an interval when the frames of U_2 interferes with the frame $\tau_{7,0}$ of U_4. We first calculate the IF for each frame of U_2 with a priority higher than P_7 in the LCM_2, then calculate the MIF for the station U_2. Since LCM_2 is 8, arrival of $\tau_{5,0}, \tau_{6,0}, \tau_{2,0}$ in $[0, LCM_2]$ are $0, 4, 5$ respectively. Thus, $M_2(t)(0, 8, P_7)$ can be obtained by calculating the max of $IF_2^0(t)(0, 8, P_7)$, $IF_2^4(t)(0, 8, P_7)$, $IF_2^5(t)(0, 8, P_7)$.

As can be seen from this example, the max operation is to pick up the uppermost line of all IF lines. It is important to note that the maximum interference time derived from MIF is larger than or equal to any in that of IF. The objective of using MIF instead of IF is to obtain a fast approximate algorithm [12].

A.4 Saturation Addition of *IF* or *MIF*

Saturation addition is the operation that adds multiple IFs, MIFs or transmission time of frames with the maximum slope equal to 1 [12], which is denoted by '⊕' in this paper. The objective of saturation addition is to add all the elements that may delay frame $\tau_{i,m}$ to a new interference function, such as $I_{all}(t)$. Then the first instant at which the slope of the function I_{all} becomes 0, is the earliest idle time (EIT) of CAN bus. In other words, it is the first time when $\tau_{i,m}$ can be transmitted to CAN bus. We use the operation $EIT(X)$ to find the first instant at which the slope of the function X becomes 0. An example of saturation addition and EIT is given in **Fig. A·2**.

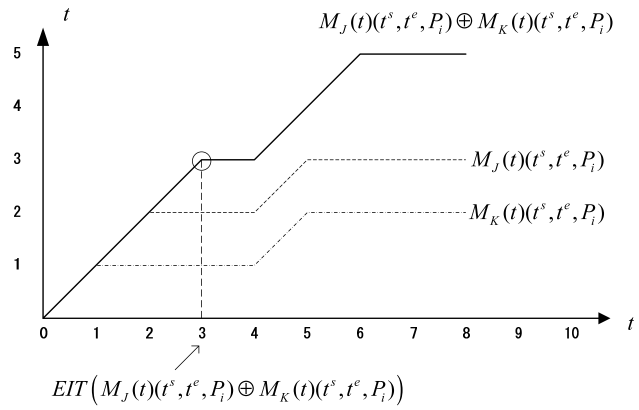
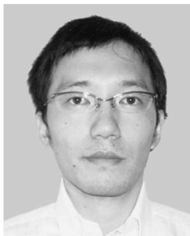


Fig. A·2 An example of saturation addition and EIT.

Yang Chen is a Ph.D student at the Department of Information Engineering, the Graduate School of Information Science, Nagoya University. He received his Master degree in Information Engineering from Xi'an Jiaotong University in 2007. His research interests include in-vehicle networks and real-time scheduling theory.



Ryo Kurachi received his bachelor and master degrees from Tokyo University of Science in 2000 and 2007. From 2000 to 2006 he was working for AISIN AW CO., LTD. Since 2007, he has been a researcher of the Center for Embedded Computing Systems, Nagoya University (NCES). His research interests include in-vehicle networks and real-time scheduling theory.

Gang Zeng is a lecturer at the Graduate School of Engineering, Nagoya University. He received his Ph.D. degree in Information Science from Chiba University in 2006. From 2006 to 2010, he was a Researcher, and then assistant professor at the Center for Embedded Computing Systems, the Graduate School of Information Science, Nagoya University. His research interests mainly include power-aware computing, real-time embedded system design. He is a member of IEEE and IPSJ.

Hiroaki Takada is a professor at the Department of Information Engineering, the Graduate School of Information Science, Nagoya University. He is also the executive director of the Center for Embedded Computing Systems (NCES). He received his Ph.D. degree in Information Science from the University of Tokyo in 1996. He was a research associate at the University of Tokyo from 1989 to 1997, and was a lecturer and then an associate professor at Toyohashi University of Technology from 1997 to 2003. His research interests include real-time operating systems, real-time scheduling theory, and embedded system design. He is a member of ACM, IEEE, IPSJ, IEICE, JSSST, and JSAE.