An Analytical Model of the CAN Bus for Online Schedulability Test

Zhenwu Shi and Fumin Zhang, Georgia Institute of Technology

***Abstract*—Controller area network (CAN) is a priority-based bus that supports real-time communication. Existing schedulability analysis for the CAN bus is peformed at the design stage, by assuming that all message information is known in advance. However, in pratice, the CAN bus may run in a dynamic environment, where complete specifications may not be available at the design stage and operational requirements may change at system run-time. In this paper, we develop an analytical model that describes the dynamics of message transmission on the CAN bus. Based on this analytical timing model, we then propose an online test that effectively checks the schedulability of the CAN bus, in the presence of online adjustments of message streams. Simulations show that the online test can accurately report the loss of scheduability on the CAN bus.**

I. Introduction

Controller area network (CAN bus) is a serial bus designed for industrial environments. It was first deployed by the automobile industry in the 1980s for decreasing wiring hardness and complexity among electronic control units in vehicles. During the past thirty years, other industries have gradually adopted the CAN bus and applied it to a variety of areas such as robotics, aircraft, medical equipment, and industrial automation. The CAN bus has an important feature of supporting real-time communication. Each message on the CAN bus is assigned a priority and the transmission of messages follows a deterministic order decided by their prorities [1]. This feature makes the CAN bus particularly suitable for systems with stringent time constraints on the communication.

Related Works The application of the CAN bus requires scheduling analysis to check if all messages can meet their deadlines. The scheduling analysis of the CAN bus has been extensively studied. In 1994, Tindell et al. proposed a basic CAN schedulability analysis [2]–[4]. This result was later recognized by Volvo Car Corporation and successfully used as the theoretical foundation for commercial CAN schedulability analysis tools [5]. However, the basic CAN schedulability analysis in [2]–[4] is based on ideal assumptions of the CAN bus that may not be supported in real applications. Since then, a lot of research has been conducted to improve the basic CAN schedulability analysis. [6]–[9] studied the effect of hardware limitations on the scheduling analysis of the CAN bus. [10]–[12] extended the basic CAN schedulability analysis to account for the transmission errors. [13], [14] studied the schedulability issues of the CAN bus when messages have offsets. [15] proposed a probalistic analysis of the response time of messages on the CAN bus. In 2007, Davis et al. revisted the basic CAN schedulability analysis and corrected some significant flaws [16].

Motivation. Existing schedulability analysis for the CAN bus is performed at the design stage, by assuming that all of message information is known in advance. This assumption used to be valid in the earlier design of embedded systems, which deliberately abstracted away properties of the physical world. However, this is not always be the case. Recent years have witnessed a growing trend for developing embedded systems that are closely integrated with the physical world. For example, Cyber Physical Systems research community has emerged, with the aim of underpinning the integration of cyber and physical elements across all application sectors [17], [18]. However, one of the major design challenges brought by this trend is that the embedded system needs to operate in a dynamic environment, where complete specifications are not possible at the design stage and/or operational requirements may change at system runtime [19]. Particularly concerning the CAN bus in embedded systems, the design challenge means that the CAN bus may frequently experience online adjustment of communication requirements, such as addition, removal, and change of message streams. To guarantee normal operation of the CAN bus in a dynamic environment, we need an online test that can check the schedulability of the CAN bus during system runtime.

Contributation. In this paper, we analytically model the dynamics of message transmission on the CAN bus through a non-block, deterministic hybrid automaton. Then, based on this analytical timing model, we propose an online test that re-evaluates the schedulability of the CAN bus. Our contribution is two manifold: (1) an analytical model of the CAN bus; and (2) an online schedulability test, which is necessary and sufficient, hence gives the least conservative schedulability test.

Organization. The remainder of this paper is organized as follows. Section II presents an overview of the CAN protocol. Section III formulates the problem. Section IV uses a hybrid automaton to analytically describe the dynamics of scheduling messages on the CAN bus. Section V proposed

The research is supported by ONR grants N00014-08-1-1007, N00014-09-1-1074, and N00014-10-10712 (YIP), and NSF grants ECCS-0841195 (CAREER), CNS-0931576, and ECCS-1056253.

Zhenwu Shi and Fumin Zhang are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332, USA zwshi, fumin@gatech.edu

an online schedulability test. Section VI uses simulations to show the effectiveness of our online schedulability test.

II. Controller Area Network

This section overviews fundamentals of the CAN protocol, with emphasis on message routing and bus arbitration, which are considered as key features of the CAN protocol. Other part of the CAN protocol can be found in technique documents avaialable online.

A. Message Routing

The CAN protocol defines a standard data message that encapsulates information transmitted between a source node and one or more receivers. As shown in Fig. 1, the data message is composed of seven fields: an SOF field, which represents the start of the message; an identifier field, which is a unique number assigned to the message; a control field, which indicates the length of the data field; a data field, which contains the information encapsulated in the message; a CRC field, which checks the integerity of the message; an ACK field, which acknowledges the reception of the message; and an EOF field, which represents the end of the message.

Figure 1. A standard message frame in CAN

The CAN protocol is a content-based protocol rather than an address-based protocol such as TCP [20]. Unlike the latter, which assigns each message an explicit destination address, the CAN protocol assigns each message a unique identifier. Based on the identifier, the CAN protocol routes messages as follows: When a node attempts to transmit a message, it broadcasts the message on the CAN bus; and each individual node receives the message from the CAN bus and, based on the identifer, decides whether or not to process the message. Such content-based protocol has two major advantages. First, a message can be destined for any number of nodes simultaneously, which increases the utilization of the CAN bus. Second, additional nodes can be added to the existing CAN bus without the necessity to reprogram all other nodes to recognize this addition, which increases the flexibility of the CAN bus.

B. Bus Arbitration

The CAN bus is a serial bus which only allows one node to transmit a message at a time. If two or more nodes attempt to transmit messages at the same time, collisions will happen on the CAN bus. The CAN protocol resolve the collisions through an arbitration scheme known as CSMA/BA (Carrier Sense Multiple Access with Bitwise Arbitration), which uses the identifier of each message as its priority and then based on priorties decides which message may be granted access to the CAN bus.

Using identifiers as priorities is enabled by the wired-AND property of the CAN bus: if multiple nodes are transmitting messages at the same time and one of the nodes transmits a logic bit “0”, the value of the bus will be “0”; and only if all of the nodes transmit a logic bit ”1” will the value of the bus be “1”. Based on this property, CSMA/BA performs arbitration as follows: (1) the arbitration starts from the first bit in the identifier field and ends at the last bit in the identifier field; (2) each node transmits the identifier of a message while monitoring the resulting bus value; and (3) if a node’s transmitted bit differs from the value of the bus, the node detects a collision and aborts its message transmission; if a node’s transmitted bit is same as the value of the bus, the node continues its message transmission. Since each message has a unique identifier, a node transmitting the last bit of the identifier without detecting a collision must be transmitting the highest priority message, and can continue transmitting the remaing part of the message. According to the above arbitration process, we can see that: (1) The logic bit “0” can always win arbitration over the logic bit ”1”, therefore the message with a lower value in the identifier field has a higher priority; (2) the highest-priority message wins arbitration without being disturbed since the transmission of lower priority messages will automatically back off and wait; and (3) the allocation of priorities to messages in identifiers makes the CAN bus particularly suitable for real-time communication.

III. Problem Formulation

We consider a system based on the CAN bus, as illustrated in Figure 2. The CAN bus is shared by multiple nodes. Each node on the CAN bus consists of three parts, where applications generate and utilize messages, host processors carry out user-defined functions, and CAN controllers implement the basic CAN protocol. Each node on the CAN bus may transmit multiple messages to other nodes.

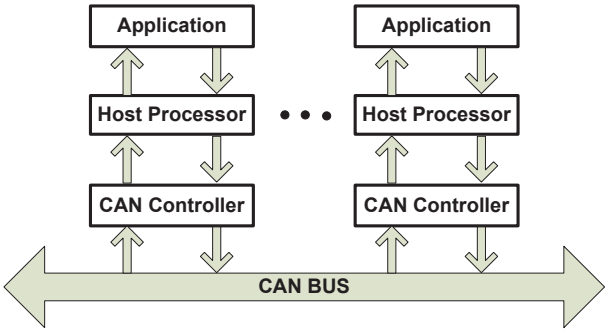


Figure 2. An example of the CAN bus

In this paper, we investigate the problem of operating the CAN bus in a dynamic environment, where communication requirement may frequently change at system run-time, such as on-line addition, removal, and change of message streams.

The online adjustment of message streams may lead to the unschedulability of the CAN bus. Thus, it is necessary to introduce a new way to re-evaluate the schedulability of the CAN bus at system run-time as follows:

Definition 3.1: An online schedulability test over a time interval $[t_a, t_b]$ checks if all message instances are able to meet their deadlines within $[t_a, t_b]$.

As the starting time increases, the time interval $[t_a, t_b]$ will slide forward. The length of the interval $t_b - t_a$ depends on how much into the future we want to perform the schedulability test. All mathematical tools developed in this paper are centered around the online schedulability test within the time interval $[t_a, t_b]$.

A. System Configuration

There are many different ways of desigining the architecture of the CAN bus. Each design may lead to a different model. In this paper, we assume that the CAN bus in Figure 2 satisfies the following conditions

1) The CAN bus is reliable such that no error exists in the transmission;
2) At each node, among all messages that are ready for transmission, the message with the highest priority will be always transmitted first.

The above conditions are conventional assumptions that have been widely used in the research community of the CAN bus. As a first step towards the analytical modeling of the CAN bus, we rely on them for our derivation. While the above conditions may not be satisfied in all application scenarios, we believe that refined modeling can be carried out under the same framework .

Moreover, we configure the CAN bus such that each node will automatically drop off transmission requests of messages that cannot meet their deadlines.

B. Model and Notations

We describe a system model for the CAN bus and several notations that will be used in the rest of the paper.

Consider a set of independent messages $\{\tau_1, \cdots, \tau_N\}$ transmitting on the CAN bus within the time interval $[t_a, t_b]$. Each message τ_n is characterized by a tuple $\{a_n^k, C_n^k, T_n^k, E_n^k, P_n^k\}$ defined as follows:

- a_n^k: the release time of the the k-th instance of τ_n;
- C_n^k: the transmission time of the k-th instance of τ_n;
- T_n^k: the inter-release time, i.e. $T_n^k = a_n^{k+1} - a_n^k$;
- E_n^k: the relative deadline of the k-th instance of τ_n;
- P_n^k: the identifier of the k-th instance of τ_n;

Note that the characteristics of τ_n is defined at the instance level. Such definition allows each instance of τ_n to have different characteristics, which makes our model applicable to not only periodic messages but also non-periodic messages.

Remark 3.2: Since the CAN bus operates in a environment where the communication requirement may frequently change, the set of messages $\{\tau_1, \cdots, \tau_N\}$ may dynamically adjust at runtime.

IV. An analytical timing model for scheduling on the CAN bus

In last section, we have established necessary notations to describe a set of messages on the CAN bus. In this section, we will develop an analytical timing model that describes the dynamics of scheduling messages on the CAN bus.

A. State Variables

To describe how message $\{\tau_1, \cdots, \tau_N\}$ are scheduled on the CAN bus from a dynamic system point of view, we introduce a state vector $Z(t) = [D(t), R(t), O(t)]$.

Definition 4.1: The first component $D(t)$ is defined as a vector $D(t) = [d_1(t), \cdots, d_N(t)]$, where $d_n(t)$, for $n = 1, 2, \cdots, N$, denotes how long after t the next instance of τ_n will be released.

Definition 4.2: The second component $R(t)$ is defined as a vector $R(t) = [r_1(t), ..., r_N(t)]$, where $r_n(t)$, for $n = 1, 2, ..., N$, denotes the remaining transmission time of the current instance of τ_n after time t.

Definition 4.3: The third component $O(t)$ is defined as a vector $O(t) = [o_1(t), ..., o_N(t)]$, where $o_n(t)$, for $n = 1, 2, ..., N$, denotes how much time has elapsed before the current instance of τ_n finishes transmission.

With the state vector well defined, we can study the dynamics of scheduling $\{\tau_1, \cdots, \tau_N\}$ on the CAN bus through the evolution of $Z(t)$.

B. Evolution of State Vector

The state vector $Z(t)$ evolve continuously most of the time, except when two "special" events happen. One special event is the arrival of a new instance. When this event happens, the value of $Z(t)$ is reset to the characteristics of the new instance. The other special event is that a message finishes transmission on the CAN bus and another message starts transmission. When this event happens, the evolution dynamics of $Z(t)$ switches discontinuously. Since the state vector $Z(t)$ exhibits both continuous and discrete dynamic behaviors, the evolution of $Z(t)$ can be described by a hybrid automaton defined as follows.

Definition 4.4: A hybrid automaton that describes the dynamics of scheduling $\{\tau_1, \cdots, \tau_N\}$ is a collection $H = \{Q, Z, F, \mathrm{Dom}, \mathrm{Edge}, \mathrm{Guard}, \mathrm{Reset}\}$ where

- $Q = \{q_0, q_1, \cdots, q_N\}$ is a set of modes, where the mode q_0 indicates that no message is being transmitted on the CAN bus and the mode q_i $(1 \leq i \leq N)$ indicates that τ_i is being transmitted on the CAN bus;

- $Z(t) = [D(t), R(t), O(t)] \in \mathbb{R}^{3N}$ is a continuous state vector as defined above;
- $F : Q \times Z \to \mathbb{R}^{3N}$ is the flow map. For each $q_i \in Q$, $F(q_i, Z)$ describes the continuous evolution of Z in the mode q_i;
- $\mathrm{Dom} : Q \to 2^Z$ is the domain of modes. For each $q_i \in Q$, $\mathrm{Dom}(q_i)$ identifies a set of Z that evolves continuously in the mode q_i;
- $\mathrm{Edge} : \mathrm{Edge} \subseteq Q \times Q$ is a set of edges. Each $(q_i, q_j) \in \mathrm{Edge}$ indicates that a discrete transition from the mode q_i to the mode q_j is possible;
- $\mathrm{Guard} : \mathrm{Edge} \to 2^Z$ is the jump condition. For each $(q_i, q_j) \in \mathrm{Edge}$, $\mathrm{Guard}(q_i, q_j)$ identifies a set of Z that can trigger a discrete transition from the mode q_i to the mode q_j;
- $\mathrm{Reset} : \mathrm{Edge} \times Z \to 2^Z$ is the reset map. For each $(q_i, q_j) \in \mathrm{Edge}$, $\mathrm{Reset}(q_i, q_j, Z)$ describes the value to which Z is reset during a discrete transition from the mode q_i to the mode q_j;

Figure 3 demonstrates a directed graph representation of the hybrid automaton H when two independent messages $\{\tau_1, \tau_2\}$ are being transmitted on the CAN bus. The graphical representation of the H with other values of N can be easily constructed using the same methodology. As shown in Figure 3, the vertices represent modes and the arrows represent edges. Within each vertex the flow map and the domain set are indicated. Along each edge the jump condition and the reset map are shown.

In the following part of this section, we will derive the expressions of F, Dom, Edge, Guard and Reset, respectively.

1) **Flow Map:** We discuss the continuous evolution of $Z(t)$ in any mode $q_i \in Q$. Since $Z(t)$ consists of three components as $Z(t) = [D(t), R(t), O(t)]$, we will discuss the continuous evolution of each component respectively. We use $\Delta t > 0$ to denote an arbitrarily small change in time.

First, we study the continuous evolution of $D(t)$ in the mode q_i. Consider an element $d_n(t) \in D(t)$. According to Definition 4.1, we know that $d_n(t)$ will continuously decrease within $[t, t + \Delta t]$, i.e.

$$d_n(t + \Delta t) = d_n(t) - \Delta t \tag{1}$$

which implies the continuous evolution of $d_n(t)$ as

$$\dot{d}_n(t) = \lim_{\Delta t \to 0} \frac{d_n(t + \Delta t) - d_n(t)}{\Delta t} = -1 \tag{2}$$

Therefore, we have the continuous evolution of $Q(t)$ in the mode q_i as

$$\dot{D}(t) = [-1, \cdots, -1]^T \tag{3}$$

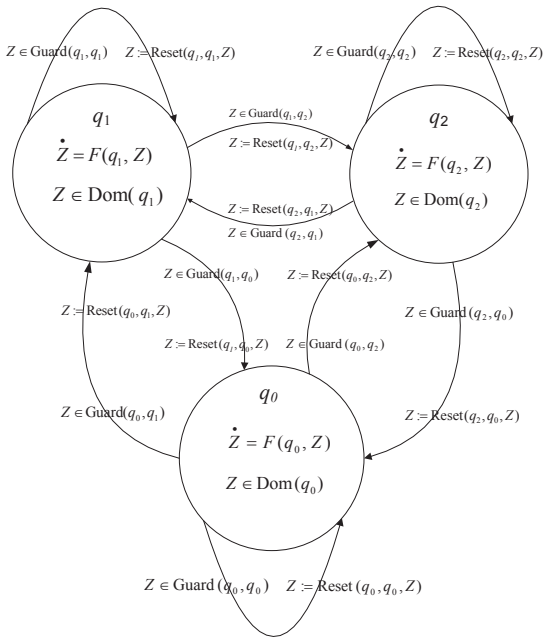


Figure 3. Graphical representation of the hybrid automaton H when $N = 2$

Next, we study the continuous evolution of $R(t)$ in the mode q_i. Consider an element $d_n(t) \in R(t)$. According to Definition 4.2, we know that $r_n(t)$ will decrease within $[t, t + \Delta t]$ when τ_n is being transmitted at time t, and keeps constant otherwise. Moreover, only τ_i can be transmitted in the mode q_i, as stated in Definition 4.4. Therefore, we have that

$$r_n(t + \Delta t) = \begin{cases} r_n(t) - \Delta t & n = i \\ r_n(t) & \text{otherwise} \end{cases} \tag{4}$$

which implies the continuous evolution of $r_n(t)$ as

$$\dot{r}_n(t) = \begin{cases} -1 & n = i \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Therefore, we can express the continuous evolution of $R(t)$ in the mode q_i as

$$\dot{R}(t) = [0, \cdots, -1, \cdots, 0]^T \tag{6}$$

where -1 is in the i-th entry of the vector.

Finally, we study the continuous evolution of $O(t)$ in the mode q_i. Consider an element $o_n(t) \in O(t)$. According to Definition 4.3, we know that the evolution of $o_n(t)$ depends on whether the current instance of τ_n has finished transmission. If the current instance of τ_n has finished transmission before time t, i.e. $r_n(t) = 0$, $o_n(t)$ will not increase within $[t, t + \Delta t]$. On the other hand, if the current instance of τ_n has not finished transmission before t, i.e. $r_n(t) > 0$, $o_n(t)$ will increase within $[t, t + \Delta t]$. By defining a function sgn such that $\mathrm{sgn}(x) = 1$ when $x > 0$, $\mathrm{sgn}(x) = 0$ when $x = 0$,

and $\mathrm{sgn}(x) = 1$ when $x < 0$, we have that

$$o_n(t + \Delta t) = o_n(t) + \mathrm{sgn}(r_n(t))\,\Delta t \tag{7}$$

which implies the continuous evolution of $o_n(t)$ as

$$\dot{o}_n(t) = \lim_{\Delta t \to 0} \frac{o_n(t + \Delta t) - o_n(t)}{\Delta t} = \mathrm{sgn}(r_n(t)) \tag{8}$$

Therefore, we have the continuous evolution of $O(t)$ in the mode q_i as

$$\dot{O}(t) = [\mathrm{sgn}(r_1(t)), \cdots, \mathrm{sgn}(r_N(t))]^T \tag{9}$$

In summary, equation (3), (6) and (9) constitute the continuous evolution of $Z(t)$ in any mode $q_i \in Q$.

2) **Domain of Modes:** For the ease of expression, we define an auxiliary variable $G(t)$ as follows

Definition 4.5: $G(t)$ is a set of indices of messages, which are active for transmission at time t.

$$G(t) = \left\{n|\ r_n(t) = C_n^k \text{ and } o_n(t) + C_n^k \le E_n^k\right\} \tag{10}$$

where k represents the index of the current instance of τ_n at time t, $r_n(t) = C_n^k$ specifies messages that have not transmitted yet, and $o_n(t) + C_n^k < E_n^k$ specifies messages that can meet their deadlines.

In the mode q_0, the state Z will continuously evolve as long as the following two conditions are both satisfied: no new instance of $\{\tau_1, \cdots, \tau_N\}$ is released and no message is active for transmission on the CAN bus. To meet the first condition, we have that

$$\min_{1 \le n \le N} \{D(t)\} > 0 \tag{11}$$

where Definition 4.1 is applied. To meet the second condition, we have that

$$\mathrm{Card}\left(G(t)\right) = 0 \tag{12}$$

where $\mathrm{Card}(\cdot)$ is a cardinality function that measures the number of elements in a set. Therefore, we have the domain of the mode q_0 as

$$\mathrm{Dom}(q_0) = \left\{ Z(t)\,|\, \min_{1 \le n \le N} \{D(t)\} > 0 \text{ and } \mathrm{Card}(G(t)) = 0 \right\} \tag{13}$$

In any other mode q_i where $1 \le i \le N$, the state Z will continuously evolve as long as the following two conditions are both satisfied: no new instance of $\{\tau_1, \cdots, \tau_N\}$ is released and τ_i is being transmitted on the CAN bus. To meet the first condition, we have that

$$\min_{1 \le n \le N} \{D(t)\} > 0 \tag{14}$$

where Definition 4.1 is applied. To meet the second condition, we have that

$$0 < r_i(t) \le C_i^k \tag{15}$$

Therefore, we have the domain of the mode q_i where $1 \le i \le N$ as

$$\mathrm{Dom}(q_i) = \left\{ Z(t)|_{\min_{1 \le n \le N} \{D(t)\} > 0 \text{ and } 0 < r_i(t) \le C_i^k} \right\} \tag{16}$$

3) **Edges and Jump Conditions:** According to the definition of Q, we know that the transition between any two modes is possible. Therefore, we have that

$$\mathrm{Edge} = Q \times Q \tag{17}$$

where an edge $(q_i, q_j) \in \mathrm{Edge}$ represents a transition from the mode q_i to the mode q_j.

For any edge $(q_i, q_j) \in \mathrm{Edge}$, we discuss the jump condition $\mathrm{Guard}(q_i, q_j)$. Our discussion on jump conditions can be classified into four cases according to different edges.

Case 1: an edge where $i = j$. This transition is triggered when any message in $\{\tau_1, \cdots, \tau_N\}$ release a new instance, i.e.

$$\mathrm{Guard}(q_i, q_i) = \{ \min_{1 \le n \le N} \{D(t)\} = 0 \} \tag{18}$$

Case 2: an edge where $i \ne j$ and $1 \le i, j \le N$. This transition is triggered when τ_i finishes transmission and τ_j starts transmission. Thus, we have

$$\mathrm{Guard}(q_i, q_j) = \{r_i(t) = 0 \text{ and } j = \operatorname*{argmin}_{n \in G(t)} P_n^k\} \tag{19}$$

where $r_i(t) = 0$ indicates that τ_i has finished transmission on the CAN bus at time t and $j = \mathrm{argmin}_{n \in G(t)} P_n^k$ indicates that τ_j has the highest priority among all messages that are active for transmission at time t.

Case 3: an edge where $1 \le i \le N$ and $j = 0$. This transition is triggered when τ_i finishes transmission and no message is active for transmission at time t. Thus, we have

$$\mathrm{Guard}(q_i, q_0) = \{r_i(t) = 0 \text{ and } \mathrm{Card}\left(G(t)\right) = 0\} \tag{20}$$

where $r_i(t) = 0$ indicates that τ_i has finished transmission at time t and $\mathrm{Card}\left(G(t)\right) = 0$ indicates that no message is active for transmission at time t.

Case 4: an edge where $i = 0$ and $1 \le j \le N$. This transition is triggered when τ_i starts its transmission after the CAN bus has been idle for a while.

$$\mathrm{Guard}(q_0, q_j) = \{\mathrm{Card}\left(G(t)\right) > 0 \text{ and } j = \operatorname*{argmin}_{n \in G(t)} P_n^k\} \tag{21}$$

where $\mathrm{Card}\left(G(t)\right) > 0$ indicates that there are messages active for transmission on the CAN bus at time t and $j = \mathrm{argmin}_{n \in G(t)} P_n^k$ indicates that τ_j has the highest priority among all messages that are active for transmission at time t.

4) **Reset Map:** We use t^+ to denote the time right after the reset.

We first discuss the reset map for an edge (q_i, q_j) where $i = j$. As discussed in equation (18), this transition happens when a new instance of $\{\tau_1, \cdots, \tau_N\}$ is released. Consider a message τ_n.

Case 1: if the new instance released at time t is not from τ_n, i.e. $d_n(t) > 0$. In this case, the state variables of τ_n hold their values during the transition, i.e.

$$\begin{gathered} \text{if } \ d_n(t) > 0, \text{we have :} \\ d_n(t^+) = d_n(t) \ \ r_n(t^+) = r_n(t) \ \ o_n(t^+) = o_n(t) \end{gathered} \tag{22}$$

Case 2: if the new instance released at time t is from τ_n, i.e. $d_n(t) = 0$. In this case, the state variables of τ_n is reset to the characteristics of the new instance.

$$\begin{gathered} \text{if } \ d_n(t) = 0, \text{we have :} \\ d_n(t^+) = T_n^{k+1} \ \ r_n(t^+) = C_n^{k+1} \ \ o_n(t^+) = 0 \end{gathered} \tag{23}$$

Applying equation (22) and (23) for $n = 1, \cdots, N$, we have the reset map $\mathrm{Reset}(q_i, q_j, Z)$ where $i = j$.

Next, we discuss the reset map for an edge (q_i, q_j), where $i \neq j$. During this transition, the state vector $Z(t)$ remains constant. Thus, $\mathrm{Reset}(q_i, q_j, Z)$ equals to an identity map:

$$\mathrm{Reset}(q_i, q_j, Z) = Z \tag{24}$$

V. Online Schedulability Test

In this section, we show how to perform an online schedulability test of the CAN bus by utilizing the analytical timing model developed in Section IV.

A. A Necessary and Sufficient Condition for Schedulability

Consider a set of messages $\{\tau_1, \cdots, \tau_N\}$ being transmitted on the CAN bus within the time interval $[t_a, t_b]$. The online schedulability test over $[t_a, t_b]$ can be decomposed to check whether each message is able to meet its deadlines within $[t_a, t_b]$. The CAN bus is schedulable within $[t_a, t_b]$ if and only if all messages are schedulable within $[t_a, t_b]$. The following theorem states the necessary and sufficient condition for the schedulability of τ_n within $[t_a, t_b]$.

Theorem 5.1: A message τ_n is schedulable within $[t_a, t_b]$ if and only if for all time points $t \in [t_a, t_b]$ such that $d_n(t) = 0$, we have that $r_n(t) = 0$.

Proof: According to Definition 4.1, we know that $d_n(t) = 0$ implies that an new instance of τ_n is released at time t. At this time point, the state variables of τ_n represent the final status of the old instance. Hence, the old instance of τ_n can meet its deadline if and only if the remaning transmission time is zero, i.e.

$$r_n(t) = 0$$

■

B. Initial State and Message Characteristics

At any time t_a, running the analytical timing model of the CAN bus within $[t_a, t_b]$ requires informaiton of the initial state and message characteristics.

First, we discuss the reconstruction of the initial state $[Q(t_a), Z(t_a)]$. For $Q(t_a)$, since messages are broadcast on the CAN bus, each node can easily detect which message is being transmitted on the CAN bus. For $Z(t_a)$, according to Definition 4.1, 4.2, and 4.3, its value depends on two types of information: characteristics of current instances of $\{\tau_1, \cdots, \tau_N\}$, and how these instances have transmitted from time of release till time t_a. The first type of information has already known at the time when an instance is released. The second type of information can be obtained from software tools that monitor and record data traffic on the CAN bus. For example, Esd Electronics, Inc. provides a monitoring tool as integral part of the driver for their CAN controllers.

Next, we consider message characteristics. In this paper, we assume that message characteristics within $[t_a, t_b]$ is known at time t_a. This assumption is reasonable as it is possible to predict a little bit ahead in the future. Also, this is the weakest assumption for performing any online schdulability test.

C. Implementation of the online schedulability test

At any time t_a, given the initial state variable and message characteristics within $[t_a, t_b]$, we can perform the online schedulability test over the time interval $[t_a, t_b]$ using Algorithm 1. This algorithm iteratively checks the schedulability of the CAN bus in the following ways: (1) within each discrete mode, it allows the state vector $Z(t)$ to continuously evolve according to the flow map F until $Z(t)$ reaches the boundary of the discrete domain and triggers a transition, as indicated by Line 8 and 9; (2) if the transition is from a mode to itself, it evaluates the schedulability of τ_n according to Theorem 5.1, as shown in Lines $11 - 15$; and (3) if the transition is between two different modes, it re-calcuates the destination mode, as indicated by Line 18.

The variable DS_n indicates the online schedulability test result of τ_n within $[t_a, t_b]$: when an instance of τ_n is schedulable, its corresponding element in DS_n equals to 0; otherwise, its corresponding element in DS_n equals to 1. A message τ_n is schedulable within $[t_a, t_b]$ if and only if all instances of τ_n that are released within $[t_a, t_b]$ are schedulable, i.e. $\max\{\mathrm{DS_n}\} = 0$. The CAN bus is schedulable within $[t_a, t_b]$ if and only if all messages are schedulable within $[t_a, t_b]$, i.e. $\max_{1 \leq n \leq N}\{\max\{\mathrm{DS_n}\}\} = 0$.

VI. Simulation

In this section, we use a simple example to demonstrate the effectiveness of our online schedulability test when

Algorithm 1: Online Schedulability Test

/*Schedulability of the CAN bus within $[t_a, t_b]$ */
Data: t_a, t_b, $Q(t_a^-)$, $Z(t_a^-)$, $\{C_n^k, T_n^k, E_n^k, P_n^k\}_{n=1}^N$
Result: $\{\mathrm{DS_n}\}_{\mathrm{n=1}}^{\mathrm{N}}$

$t = t_a$; mode $= Q(t_a^-)$;
for $n = 1$ **to** N **do**
  $\mathrm{DS_n} = [\,]$;
while $t < t_b$ **do**
  switch mode **do**
    ……;
    case n :
      while $Z(t) \in \mathrm{Dom}(q_n)$ **do**
        $Z(t) = F(q_n, Z(t))$;
      if $\min\limits_{1 \le n \le N}\{D(t)\} == 0$ **then**
        for $n = 1$ **to** N **do**
          if $d_n(t) == 0$ **and** $r_n(t) == 0$ **then**
            $\mathrm{DS_n} = [\mathrm{DS_n}\ \ 0]$;
          else if $d_n(t) == 0$ **and** $r_n(t) > 0$ **then**
            $\mathrm{DS_n} = [\mathrm{DS_n}\ \ 1]$;
        $Z(t) = \mathrm{Reset}(q_n, q_n, Z(t))$;
      else
        Re-calculate the value of mode ;
    ……;
return $\{\mathrm{DS_n}\}_{\mathrm{n=1}}^{\mathrm{N}}$;

message streams on the CAN bus frequently changes. Our methods can be easily applied to more complex examples.

At the design stage, we assume that three periodic messages $\{\tau_1, \tau_2, \tau_3\}$ start their transmission on the CAN bus simultaneously from time 0. The three messages have the following characteristics

$$\begin{aligned}
\left[\ T_1^k,\ T_2^k,\ T_3^k\ \right] &= [\ 200,\ 250,\ 300\ ]\ \mathrm{ms},\\
\left[\ C_1^k,\ C_2^k,\ C_3^k\ \right] &= [\ \ 40\ ,\ \ 50\ ,\ \ 60\ ]\ \mathrm{ms},\\
\left[\ E_1^k,\ E_2^k,\ E_3^k\ \right] &= [\ 100,\ 175,\ 200\ ]\ \mathrm{ms}.
\end{aligned}$$

Moreover, the three messages are assigned unique identifiers such that

$$P_1^k < P_2^k < P_3^k$$

which implies that τ_1 has a higher priority than τ_2, which has a higher priority than τ_3.

At system run-time, we consider two types of online communication adjustments. One is the change of message periods. We assume that τ_1 and τ_2 change their periods within $[22, 26]$s as

$$\left[\ T_1^k,\ T_2^k\ \right] = [160,\ 210]\ \mathrm{ms}.$$

The other type of online adjustment is the addition of new messages. We assume that another periodic messages τ_4 appears on the CAN bus within $[20, 26]$s. The new message has the following characteristics

$$\left[\ T_4^k,\ C_4^k,\ E_4^k\ \right] = [\ 200,\ 50,\ 200]\ \mathrm{ms},$$

Moreover, the new message is assigned a unique identifier such that

$$P_1^k < P_4^k < P_2^k < P_3^k$$

Figure 4(a) shows the transmission of four messages on the CAN bus within the time interval $[20.5, 23.5]$s. The value "0.5" indicates that the transmission of the message is blocked by other higher priority messages on the CAN bus; the value "1" indicates that the message is being transmitted on the CAN bus; and the value "0" indicates that the message finishes transmission. By closely examining Figure 4(a), we can see that two instances of τ_3 that are released at time 21s and time 22.8s have not been transmitted before their deadlines. Figure 4(a) shows the online schedulability test of the CAN bus within the time interval $[20.5, 23.5]$. As discussed in Section V, the value "1" indicates that an instance fails to meet its deadline and the value "0" indicates that an instance meets its deadline. According to Figure 4(b), we can easily see that two instances of τ_3 that are released at time 21s and time 22.8s fail to meet their deadlines. Therefore, the observation in Figure 4(b) exactly match that in Figure 4(a), which implies that our online schedulability test can accurately identify the unschedulable message instances.

VII. Conclusion

This paper proposes an online schedulability test of the CAN bus, which is based on an analytical model. The simulations show that the online schedulability test can accurately report the lost of schedulability on the CAN bus.

References

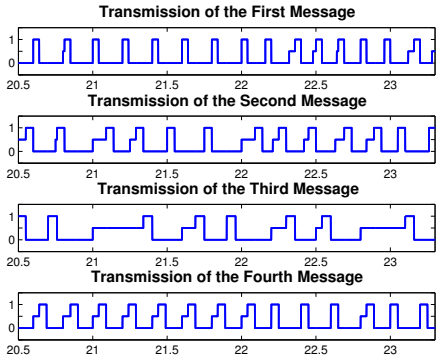
[1] R. B. Gmbh, "CAN Specification Version 2.0," Stuttgart, Germany, Tech. Rep., 1991.

[2] K. W. Tindell, H. Hansson, and A. J. Wellings, "Analysing real-time communications: controller area network (CAN)," in *Proc. of IEEE International Conference on Real-Time Systems Symposium*, Dec. 1994, pp. 259–263.

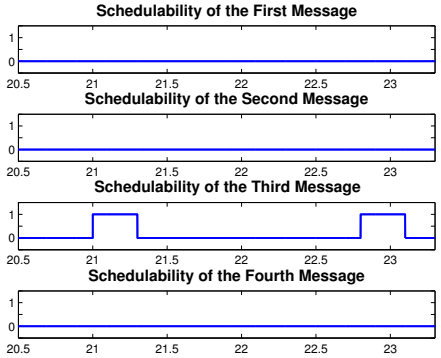
[3] K. Tindell and A. Burns, "Guarantee Message Latency on Control Area Network(CAN)," in *Proc. of International CAN Conference*, 1994, pp. 1–11.

[4] K. Tindell, A. Burns, and A. Wellings, "Calculating Controller Area Network (CAN) message response times," *Control Engineering Practice*, vol. 3, no. 8, pp. 1163–1169, 1995.

[5] L. Casparsson, A. Rajnak, K. Tindell, and P. Malmberg, "Volcano-a revolution in on-board communications," Volvo Technology Report, Tech. Rep., 1998.



(a) Transmission of Messages



(b) Online Schedulability Test Result

Figure 4. The dynamics of the CAN bus within $[20.523.3]$s

[6] M. D. Natale, "Evaluating message transmission times in Controller Area Networks without buffer preemption," in *Proc. of Brazilian Workshop on Real-Time Systems*, 2006.

[7] D. A. Khan and R. J. Bril, "Integrating hardware limitations in CAN schedulability analysis," in *Proc. of IEEE International Workshop on Factory Communication Systems*, 2010, pp. 207–210.

[8] R. I. Davis, S. Kollmann, V. Pollex, and F. Slomka, "Controller Area Network (CAN) Schedulability Analysis with FIFO queues," in *Proc. of Euromicro Conference on Real-Time Systems*, 2011, pp. 45–56.

[9] D. A. Khan, R. I. Davis, and N. Navet, "Schedulability analysis of CAN with Non-abortable Transmission requests," in *Proc. of IEEE International Conference on Emerging Technologies and Factory Automation*, Sept. 2011, pp. 1–8.

[10] I. Broster and A. Burns, "Probabilistic analysis of CAN with faults," in *Proc. of 23rd IEEE Real-Time Systems Symposium*, 2002, pp. 269–278.

[11] S. Punnekkat, H. Hansson, and C. Norstrom, "Response time analysis under errors for CAN," in *Proc. of IEEE Real-Time Technology and Applications Symposium*, 2000, pp. 258–265.

[12] I. Broster, A. Burns, and G. RodrÍguez-Navas, "Timing Analysis of Real-Time Communication under Electromagnetic Interference," *Real-Time Systems*, vol. 30, pp. 55–81, 2005.

[13] M. Grenier, L. Havet, and N. Navet, "Pushing the limits of CAN-scheduling frames with offsets provides a major performance boost," in *Proc. of European Congress Embedded Real Time Software*, 2008.

[14] P. M. Yomsi, D. Bertrand, N. Navet, and R. I. Davis, "Controller Area Network (CAN): Response Time Analysis with Offsets," in *Proc. of IEEE International Workshop on Factory Communication Systems*, 2012, pp. 43–52.

[15] H. Zeng, M. D. Natale, P. Giusto, and A. Sangiovanni-Vincentelli, "Stochastic Analysis of CAN-Based Real-Time Automotive Systems," *IEEE Transactions on Industrial Informatics*, vol. 5, no. 4, pp. 388–401, Nov. 2009.

[16] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien, "Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised," *Real-Time Systems*, vol. 35, pp. 239–272, 2007.

[17] W. Wolf, "Cyber-physical systems," *Computers*, vol. 42, no. 3, pp. 88–89, 2009.

[18] L. Sha, S. Gopalakrishnan, X. Liu, and Q. Wang, *Machine Learning in Cyber Trust Security, Privacy, and Reliability*. Springer, 2009, pp. 3–13.

[19] E. A. Lee, "Cyber Physical Systems : Design Challenges," in *Proc. of IEEE Symposium on Object Oriented Real-Time Distributed Computing*, 2008, pp. 363–369.

[20] K. Pazul, "Controller Area Network (CAN) Basics," Microchip Technology Inc., Tech. Rep., 1999.