Journal of Systems Architecture 81 (2017) 101–111

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

An optimized design of CAN FD for automotive cyber-physical systems

Xie Yong[a,b,*], Zeng Gang[c], Ryo Kurachi[d], Xie Guoqi[e], Dou Yong[b], Zhou Zhili[f]

[a] College of Computer and Information Engineering, Xiamen University of Technology, China
[b] National Laboratory for Parallel and Distributed Processing, National University of Defensive Technology, China
[c] Graduate School of Engineering, Nagoya University, Japan
[d] Graduate School of Information Science, Nagoya University, Japan
[e] College of Computer Science and Electronic Engineering, Hunan University, China
[f] School of Computer and Software, Nanjing University of Information Science and Technology, China

ARTICLE INFO

Keywords:
Automotive cyber-physical systems
Bandwidth utilization
CAN FD
Design method
Signal clustering
Schedulability

ABSTRACT

CAN with flexible data rate (CAN FD) is considered the next generation in-vehicle network standard for automotive cyber-physical systems. CAN FD supports a data phase bit-rate of up to 10 Mbps and message payload of up to 64 bytes. However, the substantial differences regarding the allowed message payloads and the heterogeneity of the signal periods indicate the need for a systematic design method to fully utilize its large transmission bandwidth. We propose an optimized design method for CAN FD to minimize bandwidth utilization while meeting the signal timing constraint. First, two slack evaluation metrics are defined for the quantitative analysis of the potential packing choices. Based on these metrics, we propose a clustering-based signal packing algorithm, and the schedulability of the signals and the packed messages are both verified. The proposed method is compared with other design methods proposed for both CAN and CAN FD. The experimental results demonstrated that our method is the most bandwidth efficient and can meet the timing constraint simultaneously.

1. Introduction

CAN is currently the most widely employed real-time network in automotive cyber-physical systems, however, its limited bandwidth (the maximum bit-rate of CAN is only 1 Mbps) cannot meet the requirement from the daily increasing bus load as the automotive industry moves towards electrification and informatization [1]. For example, new automotive features such as advanced driver assistance systems and blind-spot detection triggered the need for data bus with significantly high bandwidth. CAN with flexible data-rate (CAN FD) [2] is the next generation CAN technology proposed by Bosch in 2012. CAN FD can improve the data phase bit-rate of CAN to as much as 10 Mbps [3]. CAN FD is standardized in ISO 11898-1 and approved by ISO as a draft international standard [4]. Major semiconductor companies, such as Infineon, NXP and Renesas, already produce CAN FD controllers. Unlike other data buses with high bandwidth, such as FlexRay and Real-Time Ethernet, CAN FD can address the low bandwidth problem for CAN while retaining most of the CAN node's software and hardware, especially the physical layer [3]. This bandwidth improvement without significant modification is preferred in the automotive industry. CAN FD is considered as the next generation bus standard for automobiles.

The data phase bit-rate and the arbitration phase bit-rate of CAN FD can be configured to be different, and the data phase bit-rate of CAN FD can be increased to as high as 10 Mbps [2]. The allowed message payloads of CAN FD are 1/2/3/4/5/6/7/8/12/16/20/24/32/48/64 bytes, which can improve the bandwidth efficiency to a large extent. However, CAN only allows its bit-rate to be configured to 1 Mbps at most, and its maximal message payload can only reach 8 bytes. However, the significant difference between the allowed message payloads causes serious efficiency problems in using bandwidth for CAN FD, particularly because the signals produced by sensors and electronic control units (ECUs) are triggered with different periods and have sizes that are substantially shorter than the large allowed message payloads of CAN FD. CAN FD is also the basis of distributed safety-critical automotive functions. Thus, the schedulability of messages is mandatorily required to guarantee the safety of drivers and passengers. As a result, the advantage of CAN FD can be fully utilized only if an efficient design method that considers both bandwidth utilization and signal schedulability is employed.

1.1. Contributions

In this paper, we solve the design problem for CAN FD to minimize the bandwidth utilization while meeting the timing constraint on each

* Corresponding author at: College of Computer and Information Engineering, Xiamen University of Technology, China.
E-mail addresses: yongxie@xmut.edu.cn, yongxie@hnu.edu.cn (Y. Xie).

http://dx.doi.org/10.1016/j.sysarc.2017.10.008
Received 17 October 2016; Received in revised form 25 July 2017; Accepted 7 October 2017
Available online 13 October 2017
1383-7621/ © 2017 Elsevier B.V. All rights reserved.

signal, and this is the key to take full utilization of CAN FD's high bandwidth. The contributions of this paper are as follows:

(1) We propose two metrics to quantitatively evaluate the bandwidth slacks in messages, and these metrics are the basis of making potential packing decisions.
(2) We propose a heuristic signal packing algorithm that divides the signal set into different clusters and treats the signal cluster containing signals with equal period as the basic packing unit. The conformance of the timing constraints of the signal level is verified by conducting the schedulability analysis of the packed message set.
(3) We compare our method with other related works through experiments, and the results show that our design method is the most bandwidth efficient and can also conform to the signal level timing constraint as well.

1.2. Organization

Section 2 surveys related work on the design methods proposed for both CAN and CAN FD. Section 3 discusses the system models and the formal definition of the research problem. In Section 4, motivating examples are presented to explain the inefficiency of the existing work and demonstrate the effectiveness of our method. The details about the proposed design method are presented in Section 5. The experimental results are illustrated in Section 6 and the paper is concluded in Section 7.

2. Related work

The design of CAN FD can be divided into two sub-problems, which are signal packing and schedulability analysis. Then, we conduct a survey on the related work of the two sub-problems.

2.1. Signal packing

Signal packing of both CAN and CAN FD can be considered as a special case of the bin-packing problem, which is known as an NP-hard optimization problem [5–8], and previous works propose several heuristic algorithms to solve this problem.

Sandstrom et al. [6] sort the CAN signals by deadlines and propose the next-fit decreasing-based heuristic packing algorithm to optimize bandwidth utilization. While Saket et al. [7] sort the CAN signals with periods and propose a best-fit decreasing-based heuristic packing algorithm to optimize the bandwidth utilization with deadline constraint. Polzlbauer et al. [5] define a bandwidth utilization metric and design a next-fit decreasing-based heuristic packing algorithm to optimize bandwidth utilization for CAN, experiments verified that the algorithm is better than that of Sandstrom et al. However, comparison between the work of Saket et al [7] and Polzlbauer et al. [5] have not been conducted. Later, Polzlbauer et al. [8] present another extensibility-aware packing algorithm for CAN, the proposed algorithm is based on the simulated annealing and bandwidth utilization is its optimization objective. Bordoloi et al. [9] first address the signal packing of CAN FD, where the dynamic programming-based heuristic method is suggested to minimize the bandwidth waste. This approach uses the bandwidth waste of a message as the optimization metric, but the motivating examples explained in Section 4.3 indicate that “minimizing the bandwidth waste of packed messages” cannot “minimize the bandwidth utilization of the packed message set”. One common disadvantage of the aforementioned work is that only one signal is fed in packing each time and the packing process easily falls into the local optimum, as demonstrated in the examples in Section 4. Moreover, these works were not thoroughly compared. Urul. [10] presents another heuristic packing method for CAN FD, where signals are first packed together, and then the packed messages are disassembled and the included signals are assigned to other messages to improve bandwidth utilization. However, only the subset of signals contained in one message but not the whole set of signals with equal periods is considered, moreover, the signal schedulability is not verified.

Table 1
Comparison between the proposed design framework and the prior works.

	Employed Algorithm	Packing Unit	Schedulability Verification
Xie	integer linear programming	signal cluster	signal level
Sak[7]	best-fit decreasing	signal	message level
Pol[5]	next-fit decreasing	signal	message level
Bor[9]	dynamic programming	signal	message level
Urul[10]	best-fit decreasing	message	None

Considerable works attempted to solve the signal packing of CAN with other objectives, such as the security and real-time property. Pop et al. [11] use the offset to order the signals, and a heuristic algorithm is proposed to improve message schedulability. Zheng et al. [12] and Zhu et al. [13] integrate the design of CAN and ECU into the same optimization framework to optimize end-to-end worst-case response time (WCRT) and extensibility, respectively. Lin et al. [14] integrate signal packing and task allocation to minimize end-to-end WCRT. Xie et al. [15] propose a security-aware signal packing algorithm for CAN that employs mixed-integer linear programming method to improve bandwidth utilization while meeting both timing and security constraints.

2.2. Schedulability analysis

Tindell et al. [16,17] adapt and apply the research results from fixed priority scheduling for a single processor system into the scheduling of CAN messages and suggest assigning priorities in deadline minus jitter monotonic priority order (DJMPO). Later Davis et al. [18] correct an important flaws in the schedulability analysis described above; the researchers demonstrate that DJMPO is not optimal for CAN, and the Audsley's optimal priority assignment (OPA) algorithm [19] should be used instead. Further, Davis and Burns [20] suggest that robust priority ordering can tolerate the errors on a bus. Schmidt [21] proposes another robust priority assignment method for CAN, which can accommodate future new messages into the existing message set. Davis et al. [22] recently tackles the priority assignment problem of CAN message when some message priorities are already fixed. However, the above studies only analyze the schedulability of the message level.

An optimized design method for CAN FD is proposed, which considers both bandwidth utilization and timing constraint. For the signal packing process, a signal cluster but not a signal is inputted for packing each time. This approach increases the level of the packing unit; thus, it can find a bandwidth-efficient packing result. After the packing process, the schedulability analysis is conducted on the packed message set to verify if the signals can meet the timing constraints. The following table describes differences between our method and that of the prior works.

3. System Model and Problem Formulation

3.1. Signal Model

Fig. 1 presents a typical automotive cyber-physical system where several ECUs are interconnected by a CAN FD bus. ECU set is indicated as $ECU = \{E_1, E_2, \ldots E_k, \ldots E_{|ECU|}\}$ where $|ECU|$ indicates the number of ECUs. A signal set exists in each ECU. For example, the signal set in E_k is indicated as $S_k = \{s_{k,1}, s_{k,2}, \ldots s_{k,i}, \ldots, s_{k,|S_k|}\}$ where $|S_k|$ represents the number of signals in S_k. As the packing is only needs to be done for signals belonging to the same ECU, we omit the subscript k in a signal's notation for simplicity and clarity, s_i is used to indicate the signal for the following part of this paper. The real-time properties of signals are described with a 3-tuple, s_i: $\{t_i, z_i, d_i\}$, which indicates the period (in μs),

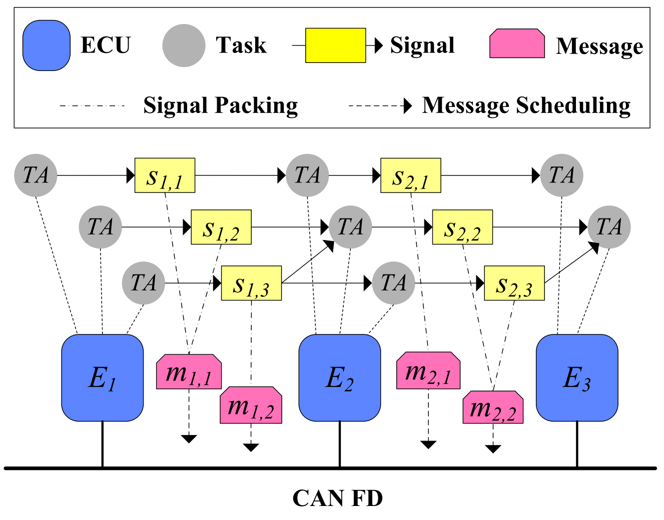


Fig. 1. Design of CAN FD for automotive cyber-physical systems.

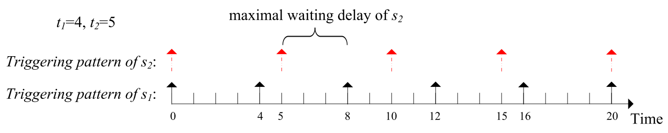

t_1=4, t_2=5
maximal waiting delay of s_2
Triggering pattern of s_2:
Triggering pattern of s_1:
0
4
5
8
10
12
15
16
20
Time

Fig. 2. Triggering patterns of two sample signals.

size (in bit) and deadline (in μs) of s_i, respectively. We assume that period and size are given for the signals, and deadline is initially set equal to its period [6,7].

To improve bandwidth utilization, we allow signals with different periods to be packed into the same message [5,9,10]; however, the oversampling of signals with big period occurs. Thus, the signal's deadline needs to be updated according to the packing result. Fig. 2 indicates the triggering patterns of s_1 and s_2, and we assume that they are packed into the same message m_1. The period of m_1 is equal to the smallest period of the included signals, which is 4 in this example. As a result, there is an offset between the release points of m_1 and s_2; d_2 needs to be updated because some instances of s_2 need time before the instances of m_1 becomes ready for transmission. The maximum waiting delay (indicated as w_i) for the instances of s_i is calculated as follows:

$$w_i = \min(t_i, t_{i'}) - \gcd(t_i, t_{i'}), \;\; s_i \in m_j \wedge \forall\, s_{i'} \in m_j \tag{1}$$

In the preceding examples, $w_1=0$, $w_2=3$. As a result, the final deadline of s_i after packing is calculated as

$$d_i = \min(t_i, t_{i'}) - w_i, \; s_i \in m_j \wedge \forall\, s_{i'} \in m_j \tag{2}$$

3.2. Message model

According to the specification of CAN FD, signals in each ECU must be packed into messages before they can be transmitted on a bus, and the number of message to be generated depends on the employed signal-packing algorithm. We assume that the packed message set is represented as $M_k = \{m_{k,1}, m_{k,2}, \ldots m_{k,j}, \ldots, m_{k,|M|}\}$, where $|M|$ indicates the number of packed message from signal set S_k of ECU E_k. Fig. 1 shows that the signal set contained in E_1 is $S_1 = \{s_{1,1}, s_{1,2}, s_{1,3}\}$, and the packed message set is $M_1 = \{m_{1,1}, m_{1,2}\}$, where $m_{1,1} = \{s_{1,1}, s_{1,2}\}$, $m_{1,2} = \{s_{1,3}\}$. We also omit subscript k of the message notation for simplicity and clarity. Real-time properties of m_j are described with a 5-tuple: $M_j = \{T_j, Z_j, P_j, C_j, D_j\}$, which indicate the period (in μs), size (in bit), payload (in byte), transmission time (in μs) and deadline (in μs) of m_j, respectively.

T_j and Z_j of m_j is calculated using the following equations:

$$T_j = \min(t_i), \;\; \forall\, s_i \in m_j \tag{3}$$

$$Z_j = \sum_{\forall s_i \in m_j} z_i \tag{4}$$

The allowed payloads of CAN FD message are 1/2/3/4/5/6/7/8/12/16/20/24/32/48/64 bytes. P_j is used to indicate the actual payload of m_j, and it can be calculated using Eq. (5).

$$P_j = \begin{cases} \lceil \frac{Z_j}{8} \rceil, & 0 \le Z_j \le 64 \\ 12, & 64 < Z_j \le 96 \\ 16, & 96 < Z_j \le 128 \\ 20, & 128 < Z_j \le 160 \\ 24, & 160 < Z_j \le 196 \\ 32, & 196 < Z_j \le 256 \\ 48, & 256 < Z_j \le 384 \\ 64, & 384 < Z_j \le 512 \end{cases}. \tag{5}$$

We assume that signals are packed according to CAN FD specification [2], although the stuff bits of the messages depend on their actual payload. We assume, for simplicity, that the maximum number of stuff bits is transmitted in each message. Thus, C_j is calculated as follows [9]:

$$C_j = 32 \times \tau_{arb} + \left(28 + 5 \times \left\lceil \frac{P_j - 16}{64} \right\rceil + 10 \times P_j \right) \times \tau_{bit} \tag{6}$$

where τ_{arb} indicates the time to transmit one bit during the arbitration phase, and τ_{bit} denotes the time to transmit one bit during the data phase. For example, if the arbitration phase bit-rate is 500 kbps and the data phase bit-rate is 2 Mbps, then $\tau_{arb} = 2$ μs and $\tau_{bit} = 0.5$ μs.

Based on the preceding analysis, the bandwidth utilization of m_j is calculated as follows:

$$U_j = \frac{C_j}{T_j}. \tag{7}$$

3.3. Problem formulation

In designing CAN FD, we aim to minimize bandwidth utilization while conforming to the timing constraints of signals. This optimization problem is formalized as:

$$\text{Minimize} \; \sum_{j=1}^{|M|} U_j$$

where three constraints must to be met: (a) Z_j cannot exceeds the allowed maximal payload of CAN FD message (64 bytes); thus, it needs to conform to the constraint

$$Z_j \le 512. \tag{a}$$

(b) Each signal is exactly packed into one message; thus, the packing of s_i needs to conform to the constraint

$$\sum_{j=1}^{|M|} \chi_{i,j} = 1. \tag{b}$$

where the binary variable $\chi_{i,\,j}$ indicates if s_i is packed into m_j or not. If that is true, $\chi_{i,\,j}=1$; otherwise, $\chi_{i,\,j}=0$. (c) Signals must be transmitted to the destination ECU before their deadline. As a result, r_i needs to meet the following constraint:

$$r_i \le d_i \tag{c}$$

where r_i indicates the WCRT of s_i, and it is equal to that of the message it belongs to. We employ a previous method [18] to calculate the message WCRT Table 1.

4. Motivating Examples

Several examples are presented to demonstrate the effectiveness of the proposed method. The two example signal sets are described in Tables 2 and 3. We assume that the arbitration phase bit-rate is 500 kbps and the data phase bit-rate is 2 Mbps. Thus, $\tau_{arb} = 2$ μs and $\tau_{bit} = 0.5$ μs.

4.1. Execution result of the methods proposed in [5,7] for the first example signal set

There are some differences about CAN and CAN FD, for example, CAN FD allows message to have much larger payload size, and this will lead to bigger design space for the design of CAN FD. However, the design problem of CAN and CAN FD is basically the same, which are signal packing, priority assignment, and schedulability analysis for packed messages. The proposed heuristic methods [5,7] are representative works to address the design problem of CAN, and we compared our method with the above mentioned heuristic methods by modifying some of their features (such as the calculation about message payload and message transmission time) according to the CAN FD specification to show if these heuristic methods are good for CAN FD or not.

According to the algorithms proposed in [5,7], one signal is inputted for packing each time. For the given example, message m_1 is first generated to accommodate s_1, and then s_2 is inputted for packing. Two packing possibilities exist for s_2, which are packing s_2 into existing m_1 or packing s_2 into a new null message; thus, we need to analyze the bandwidth utilization for these two possibilities to determine the better possibility.

Strategy 1: Pack s_2 into a new null message m_2; thus, the packed message set is $M = \{m_1, m_2\}$.

The payload, transmission time, and bandwidth utilization of m_1 are calculated as follows:

$P_1 = \lceil\frac{30}{8}\rceil = 4, C_1 = 32 \times 2 + \left(28 + 5 \times \lceil\frac{4-16}{64}\rceil + 10 \times 4\right) \times 0.5 = 98,$
$U_1 = \frac{98}{1000} = 0.098.$

The payload, transmission time, and bandwidth utilization of m_2 are calculated as follows:

$P_2 = \lceil\frac{2}{8}\rceil = 1,\ C_2 = 32 \times 2 + \left(28 + 5 \times \lceil\frac{1-16}{64}\rceil + 10 \times 1\right) \times 0.5 = 83,$
$U_2 = \frac{83}{2000} = 0.0415.$

As described above, the bandwidth utilization of the packed message set is $0.098 + 0.0415 = 0.1395$.

Strategy 2: Pack s_2 into the existing message m_1; thus, the packed message set is $M = \{m_1\}$.

The payload, transmission time, and bandwidth utilization of m_1 are calculated as follows:

$P_1 = \lceil\frac{30+2}{8}\rceil = 4,$
$C_1 = 32 \times 2 + \left(28 + 5 \times \lceil\frac{4-16}{64}\rceil + 10 \times 4\right) \times 0.5 = 98,$
$U_1 = \frac{98}{1000} = 0.098.$

As described, the bandwidth utilization of the packed message set is 0.098. Based on the preceding analysis, strategy 2 is better. Thus, the packed message set is $M = \{m_1\}$, where $m_1 = \{s_1, s_2\}$. Next, s_3 is inputted for packing. The following calculations indicate the bandwidth utilization of the two possible packing strategies.

Strategy 1: Pack s_3 into a new null message m_2; thus, the packed message set is $M = \{m_1, m_2\}$.

The payload, transmission time, and bandwidth utilization of m_2 are calculated as follows:

$P_2 = \lceil\frac{2}{8}\rceil = 1,\ C_2 = 32 \times 2 + \left(28 + 5 \times \lceil\frac{1-16}{64}\rceil + 10 \times 1\right) \times 0.5 = 83,$
$U_2 = \frac{83}{8000} = 0.010375.$

As described, the bandwidth utilization of the packed message set is $0.098 + 0.010375 = 0.108375$.

Strategy 2: Pack s_3 into the existing message m_1; thus, the packed message set is $M = \{m_1\}$.

The payload, transmission time, and bandwidth utilization of m_1 are calculated as follows:

$P_1 = \lceil\frac{32+2}{8}\rceil = 5,$
$C_1 = 32 \times 2 + \left(28 + 5 \times \lceil\frac{5-16}{64}\rceil + 10 \times 5\right) \times 0.5 = 103,$
$U_1 = \frac{103}{1000} = 0.103.$

As described, the bandwidth utilization of the packed message set is 0.103. Based on the preceding analysis, strategy 2 is better, Therefore, the packed message set is $M = \{m_1\}$, where $m_1 = \{s_1, s_2, s_3\}$. Finally, we also find that packing s_4 into the existing m_1 is bandwidth efficient. As a result, the final packing result of this example signal set is $M = \{m_1\}$, where $m_1 = \{s_1, s_2, s_3, s_4\}$.

4.2. Execution result of the method proposed in [10] for the first example signal set

According to the [10], signals with equal periods are initially packed together. As a result, the initial packing result of the example signal set is $M = \{m_1, m_2, m_3\}$, where $m_1 = \{s_1\}$, $m_2 = \{s_2\}$, $m_3 = \{s_3, s_4\}$. Then, all packed messages are iterated according to decreasing period and increasing size to determine if any existing messages can be disassembled to improve bandwidth utilization. For packed message set M, the process starts from m_3 to conduct heuristic searching. According to the following analysis, m_3 can be disassembled to improve bandwidth utilization and the included signals are moved to m_2.

Strategy 1: m_2 and m_3 coexist in the packed message set M, where $M = \{m_1, m_2, m_3\}$, $m_1 = \{s_1\}$, $m_2 = \{s_2\}$, $m_3 = \{s_3, s_4\}$.

The payload, transmission time, and bandwidth utilization of original m_2 are calculated as follows:

$P_2 = \lceil\frac{2}{8}\rceil = 1,\ C_2 = 32 \times 2 + \left(28 + 5 \times \lceil\frac{1-16}{64}\rceil + 10 \times 1\right) \times 0.5 = 83,$
$U_2 = \frac{83}{2000} = 0.0415.$

The payload, transmission time, and bandwidth utilization of m_3 are calculated as follows:

$P_3 = \lceil\frac{24}{8}\rceil = 3, C_3 = 32 \times 2 + \left(28 + 5 \times \lceil\frac{3-16}{64}\rceil + 10 \times 3\right) \times 0.5 = 93,$
$U_3 = \frac{93}{8000} = 0.01175.$

Thus, the sum bandwidth utilization of m_2 and m_3 is $0.0415 + 0.011625 = 0.053125$.

Strategy 2: Disassemble m_3 and move the included signals into m_2; the packed message set $M = \{m_1, m_2\}$, where $m_1 = \{s_1\}$, $m_2 = \{s_2, s_3, s_4\}$.

The payload, transmission time, and bandwidth utilization of updated m_2 are calculated as follows:

$P_2 = \lceil\frac{24+2}{8}\rceil = 4,$
$C_2 = 32 \times 2 + \left(28 + 5 \times \lceil\frac{4-16}{64}\rceil + 10 \times 4\right) \times 0.5 = 98,$
$U_2 = \frac{98}{2000} = 0.049.$

Based on the preceding analysis, disassembling m_3 can improve bandwidth utilization. As a result, the packed message set is updated to $M = \{m_1, m_2\}$, where $m_1 = \{s_1\}$, $m_2 = \{s_2, s_3, s_4\}$. Evaluation is continued if m_2 needs to be disassembled to improve bandwidth utilization further.

Strategy 1: m_1 and the updated m_2 coexist in the packed message set, where $M = \{m_1, m_2\}$, and $m_1 = \{s_1\}$, $m_2 = \{s_2, s_3, s_4\}$.

The payload, transmission time, and bandwidth utilization of the original m_1 are calculated as follows:

$P_1 = \lceil\frac{30}{8}\rceil = 4, C_1 = 32 \times 2 + \left(28 + 5 \times \lceil\frac{4-16}{64}\rceil + 10 \times 4\right) \times 0.5 = 98,$
$U_1 = \frac{98}{1000} = 0.098.$

Thus, the sum bandwidth utilization of m_1 and the updated m_2 is $0.098 + 0.049 = 0.147$.

Strategy 2: Disassemble the updated m_2 and move the included signals into m_1, where the packed message set is $M = \{m_1\}$, $m_1 = \{s_1, s_2, s_3, s_4\}$.

The payload, transmission time, and bandwidth utilization of the updated m_1 are calculated as follows: $P_1 = \lceil\frac{56}{8}\rceil = 7,$
$C_1 = 32 \times 2 + \left(28 + 5 \times \lceil\frac{7-16}{64}\rceil + 10 \times 7\right) \times 0.5 = 113,$

$U_1 = \frac{113}{1000} = 0.113$.

Based on the preceding analysis, disassembling m_2 is bandwidth efficient. As a result, the final packed message set is $M = \{m_1\}$, where $m_1 = \{s_1, s_2, s_3, s_4\}$.

Although the previous work [10] first considers all the signals with an equal period, the next stage of disassembling the packed messages to further improve bandwidth utilization is conducted at the message level. But Section 4.2 indicates that this step will possibly lead the packing result to the local optimum.

4.3. Execution result of the method proposed in [9] for the second example signal set

The searching space of the previous method [9] is excessively large for the sample signal set given in Table 2; thus, we changed the signal size from bit to byte by applying the ceiling function. The obtained signal set is given in Table 3.

The bandwidth overhead of the packed message m_j is calculated as follows [9]:

$$OH(AP_j) = 32 + \left(28 + 5 \times \left\lceil \frac{AP_j - 16}{64} \right\rceil + 2 \times AP_j\right). \tag{8}$$

where AP_j indicate the allowed payload of m_j. The bandwidth waste of m_j is calculated as follows:

$$BW_j(AP_j) = \frac{(8 \times AP_j - Z_j) + OH(AP_j)}{T_j} \tag{9}$$

We give the following 2 DP (dynamic programming) tables to show the details of the execution of Algorithm 1 in [9], where the packing result is attempted to be searched with the minimal bandwidth waste when AP is set as 1 and 9. Row i represents the signal s_i that can be packed into the message, and column j denotes the possible size of the packed messages. The three values in each element of the table indicate the bandwidth waste, size, and period of the packed message, respectively. The bold element in each table indicates the packing result with the minimal bandwidth waste. Further details on its algorithm are described in [9].

After the searching through all the DP tables, the minimal bandwidth waste is determined when $AP = 1$ and s_3 is packing into a message. As a result, the packed message set is $M = \{m_1\}$, where $m_1 = \{s_3\}$. The signal set is updated to $S = \{s_1, s_2, s_4\}$, and Algorithm 1 of [9] is used again to find the next packing with the minimal bandwidth waste. The final packed message set of the second example signal set is $M = \{m_1, m_2, m_3, m_4\}$, where $m_1 = \{s_3\}$, $m_2 = \{s_4\}$, $m_3 = \{s_2\}$, $m_4 = \{s_1\}$. It is obvious that this packing result is not bandwidth efficient. As a result, the minimization of bandwidth waste in the packed messages will not lead to the minimization of the bandwidth utilization of the packed message set Tables 4, 5.

4.4. Execution result of the proposed method for the first example signal set

In our approach, the signal set is first divided into clusters, where signals with equal period belong to the same cluster. This process is the same as the method proposed in [10]. As a result, S is divided into three signal clusters, which are $SC_1 = \{s_1\}$, $SC_2 = \{s_2\}$, $SC_3 = \{s_3, s_4\}$, and $S = SC_1 \cup SC_2 \cup SC_3$. We input signal clusters for packing in order of increasing period. The packing of SC_1 and SC_2 is the same as that in previous studies [5,7], and the packing result is $M = \{m_1\}$, where $m_1 = \{s_1, s_2\}$. However, when SC_3 is inputted for packing, the space to accommodate SC_3 in m_1 is sufficient, and we need to evaluate whether SC_3 is packed into existing m_1 or a new null message. The bandwidth utilization analysis for the preceding two possible packing strategies is described as follows:

Strategy 1: Pack SC_3 into a new null message m_2; thus, the packed message set is $M = \{m_1, m_2\}$.

The payload, transmission time, and bandwidth utilization of m_2 are calculated as: $P_2 = \lceil\frac{24}{8}\rceil = 3$, $C_2 = 32 \times 2 + \left(28 + 5 \times \left\lceil\frac{3-16}{64}\right\rceil + 10 \times 3\right) \times 0.5 = 93$, $U_2 = \frac{93}{8000} = 0.011625$.

As described, the bandwidth utilization of the packed message set is $0.098 + 0.011625 = 0.109625$.

Strategy 2: Pack SC_3 into the existing m_1; thus, the packed message set is $M = \{m_1\}$.

The payload, transmission time, and bandwidth utilization of m_1 are calculated as follows: $P_1 = \lceil\frac{32+24}{8}\rceil = 7$, $C_1 = 32 \times 2 + \left(28 + 5 \times \lceil\frac{7-16}{64}\rceil + 10 \times 7\right) \times 0.5 = 113$, $U_1 = \frac{113}{1000} = 0.113$.

The analysis indicates that strategy 1 is better. The packed message set is $M = \{m_1, m_2\}$, where $m_1 = \{s_1, s_2\}$, $m_2 = \{s_3, s_4\}$. Thus, the bandwidth utilization of the obtained message set is lower than that of the packed message set from previous works [5,7,10]. When the proposed approach is applied for the second signal set, the packed message set is $M = \{m_1, m_2\}$, where $m_1 = \{s_1, s_2, s_3\}$, $m_2 = \{s_4\}$. The bandwidth utilization of the obtained message set is lower than that of the packed message set from previous works [9]. As a result, the strategy of packing one signal each time easily leads to the packing result falling into the local optimum, and the bandwidth efficiency of packing the signal cluster each time is higher.

Using the preceding examples, we described the main differences between our approach and four existing methods for CAN and CAN FD. For the utilized signal sets, the proposed approach has a more bandwidth-efficient design result than all the other methods. Then, we provide a detailed description of our design method.

5. Proposed method

We propose an optimized design method for CAN FD. Before specifically describing the suggested method, we first define two bandwidth slack evaluation metrics, which are the basis for the heuristic search of the possible packing choices.

5.1. Definitions of slack bandwidth evaluation metrics

According to the CAN FD specification, message payload is in byte. However, signal size is in bit. Thus, there is possibly some slack bits may exist inside the actual message payload. Especially as the allowed payloads are quite large for CAN FD, such as 48 bytes and 64 bytes. The left slack bits are possibly large if no efficient design method is applied. If these slack bits are left unused, the corresponding bandwidth is wasted.

Definition 1 (*Payload level slack bits, PLSB*). The slack bits left inside the actual message payload are defined as the payload level slack bits: *PLSB*.

As Eq. (5) shows that the actual payload of the packed messages is calculated with a piecewise function. Thus, the $PLSB_j$ of m_j can be calculated as follows:

$$PLSB_j = 8 \times P_j - Z_j \tag{10}$$

Figure 3 shows an example of a packed message. The message size Z_j is 67 bits; thus, actual payload P_j is 12 bytes, and $PLSB_j$ is 29 bits. The maximum allowed payload for CAN FD message is 64 bytes. During the packing process, we need to monitor the payload status of the packed message in real time to assess if the space for other signals is sufficient. Thus, we provide the following definition to describe the left slack bits inside the packed message before the actual payload reaches its limit.

Definition 2 (*Message level slack bits, MLSB*). Before the actual message payload reaches its limit of 64 bytes, the slack bits left inside the message except the actual message payload is defined as the message level slack bits: *MLSB*.

The $MLSB_j$ of m_j is calculated as follows:

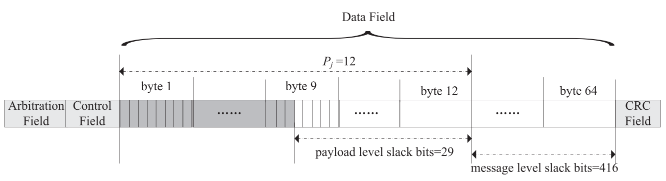


Fig. 3. Example of a packed message.

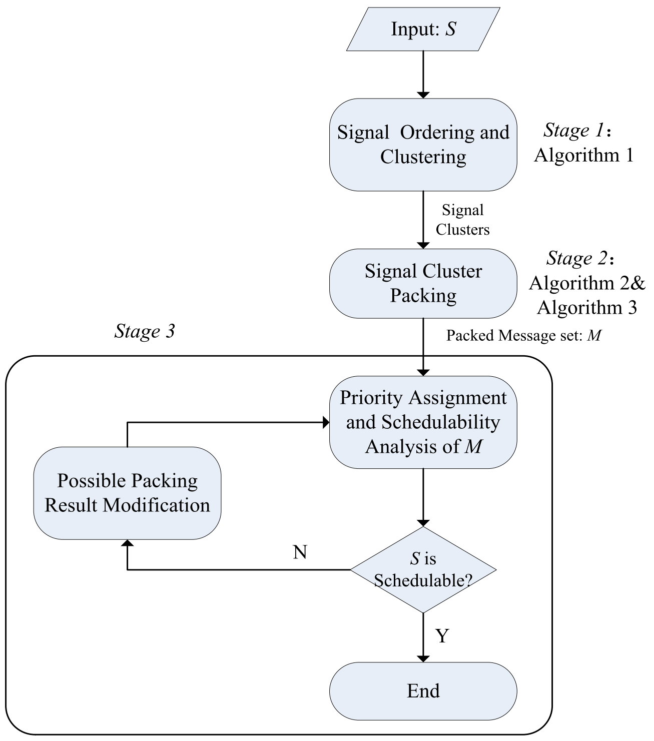

Input: S
Signal Ordering and Clustering
Stage 1: Algorithm 1
Signal Clusters
Signal Cluster Packing
Stage 2: Algorithm 2& Algorithm 3
Packed Message set: M
Stage 3
Priority Assignment and Schedulability Analysis of M
Possible Packing Result Modification
N
S is Schedulable?
Y
End

Fig. 4. Proposed design method for CAN FD.

$$MLSB_j = (64 - P_j) \times 8 \tag{11}$$

Fig. 3 shows that P_j is 12 bytes; thus, $MLSB_j$ is 416 bits.

Based on the preceding two metrics, we can quantitatively evaluate the slack bits in the packed messages, which form a solid foundation for the heuristic packing process.

5.2. The proposed design method for CAN FD

Fig. 4 shows that the proposed design method has three stages: (1) during the preparation stage, the signals are clustered and ordered, and the obtained signal clusters are inputted to the next stage for packing; (2) during the packing stage, the inputted signal clusters are packed into messages, and packed message set M is inputted to the next stage for schedulability verification; (3) during the schedulability verification stage, priorities are assigned to the messages in M and schedulability analysis is conducted to assess if the signals are schedulable or not. Then, we provide a detailed description of the aforementioned stages.

Stage 1: Signal ordering and clustering

In this stage, signal set S is the input of Algorithm 1. Unlike [5,7,9] that input one signal for packing each time, we first divide S into several signal clusters containing signals with equal period and input the signal cluster but not a single signal for packing. We increase the level of the packing unit by inputting a cluster of signals for packing. Thus, the heuristic algorithm finds bandwidth-efficient packing result. A detailed description of Algorithm 1 is as follows:

For Algorithm 1, we first call the classic merge_sort method [23] to categorize all the signals in S in order of increasing period (line 1). For Loop is executed to divide the ordered signals into signal clusters (lines 4∼10). The time complexity of this algorithm is O($n$log$n$+n).

Stage 2: Signal cluster packing

In this stage, the obtained signal clusters from Algorithm 1 are inputted for packing, and the details are given in Algorithm 2.

In Algorithm 2, a for loop takes one signal cluster for packing each time (lines 2-41). If the current SC_k is with the smallest period, all its included signals are packed into null messages directly (lines 3-8). To minimize bandwidth utilization, we formulate the signal packing problem into an integer linear programming (ILP) problem, and the CPLEX tool from IBM is employed to solve the ILP formulation. The details of the ILP formulation are described in Algorithm 3.

We assume that $s_i \in SC_k$, and the binary variable $\chi_{i,\,j}$ indicates if s_i is packed into m_j or not.

We use obj_size_j to represent the available slack bits in m_j to accommodate signals of SC_k. For the function call of ILP_Pack(SC_k, m_{null}, 0), obj_size_j is equal to the maximal allowed payload of the CAN FD message; for the function call of ILP_Pack(SC_k, m_{exist}, 1), obj_size_j is equal to the sum of $PLSB_j$ and $MLSB_j$; for the function call of ILP_Pack(SC_k, m_{exist}, 2), obj_size_j is equal to the $PLSB_j$. We use sum_size_j to indicate the sum size of the packed signals.

If SC_k is not with the smallest period, a while loop packs SC_k (lines 9-39). The proposed method needs to assess whether the slack bits in the existing messages of M can accommodate all signals of SC_k or not (line 11). If that is true, two packing possibilities exist for SC_k, which are packing it into existing messages of M or new null messages. As a result, we need to analyze the bandwidth utilization for the two packing possibilities and decide the possibility that is more bandwidth efficient (line 12). If packing SC_k into null messages is better, a new null message is generated and some signals of SC_k are packed into it by calling the Algorithm 3. $PLSB$ and $MLSB$ of the newly packed messages are analyzed, and M and SC_k are updated correspondingly (lines 13-16). Otherwise, all the signals of SC_k are packed into existing messages, the $PLSB$ and $MLSB$ of the enlarged messages are updated, and the processing continues for the next signal cluster (lines 18-24).

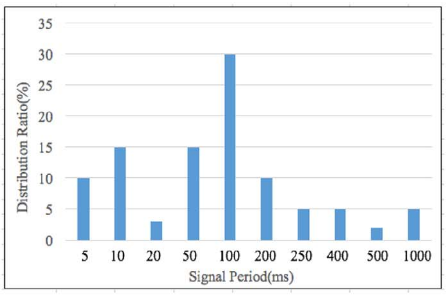
If the slack bits in the existing messages of M cannot accommodate all the signals of SC_k, some signals of SC_k are first packed into the $PLSB$ of the existing messages by using Algorithm 3 (lines 26-29). $PLSB$ and $MLSB$ of the enlarged messages are updated, and the packed signals are deleted from SC_k. If SC_k is not null, the signals of SC_k are packed into null messages by using Algorithm 3 (lines 31-35). The $PLSB$ and $MLSB$ of the newly packed message are analyzed, and M and SC_k are updated correspondingly. The output of Algorithm 2 is the packed message set M with the minimal bandwidth utilization, and next M is inputted to the next stage for schedulability analysis.

Stage 3: Schedulability analysis

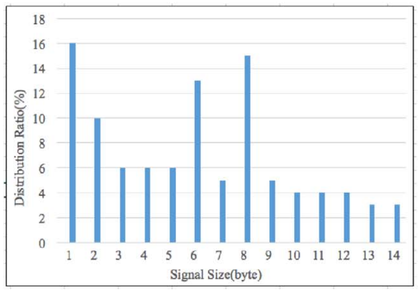
In this stage, OPA method [19] and schedulability analysis method proposed by Davis et al. [18] are utilized to analyze the WCRT for the messages of M. The schedulability of each signal can be verified accordingly with the addition of Eqs. (1)–(3) in page 5. If some signals are not schedulable, the packing result must be modified to improve the schedulability of S. For example, the packed messages containing signals with different periods can be disassembled in order of increasing period. The reason for this approach is to trade the bandwidth for schedulability, because the packing of signals with different periods into the same message will cause the oversampling of the signals with relatively large period.

6. Experiments

The comparison experiments are conducted on a OS X(v10.11) machine running on a 1.8 GHz Intel i7 core with 4 GB main memory, and two different kinds of signal sets are employed to verify the effectiveness of the proposed method. The first kind of signal sets is generated with the same approach as the previous work [9], where the periods and the sizes are uniformly distributed between 100 and 5000 ms, and between 1 and 14 bytes, respectively. The second kind of

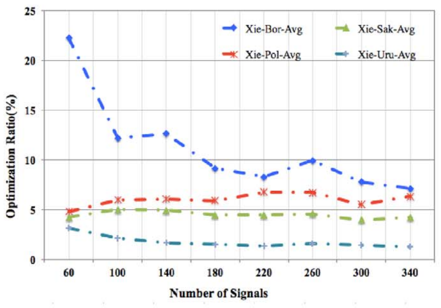


(a) Period distribution of the signal set

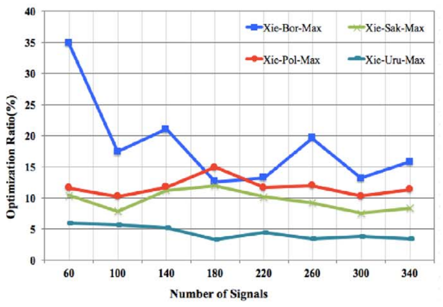


(b) Size distribution of the of signal set

Fig. 5. Period and size distributions of the second kind of signal sets.

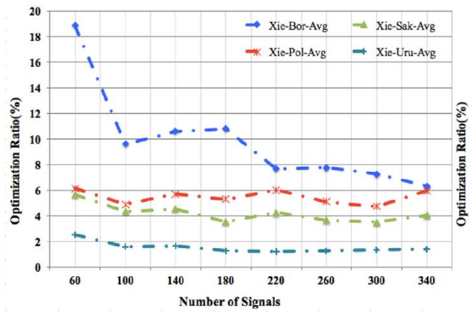


(a) Average optimization ratio

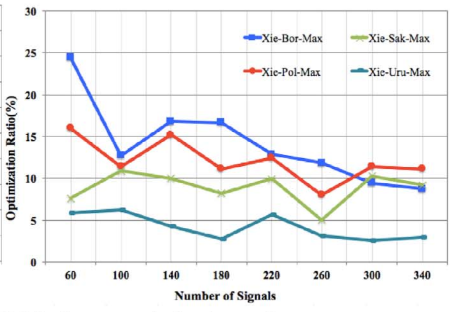


(b) Maximum optimization ratio

Fig. 6. Optimization ratio of bandwidth utilization based on the first kind of signal sets.

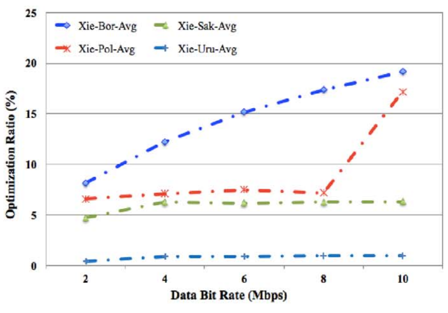


(a) Average optimization ratio

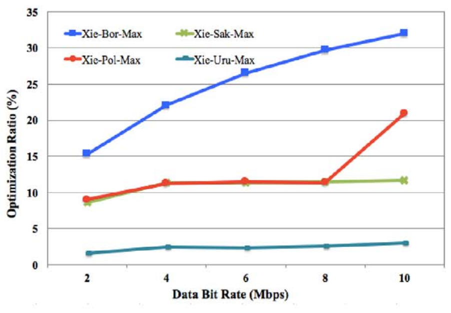


(b) Maximum optimization ratio

Fig. 7. Optimization ratio of bandwidth utilization based on the second kind of signal sets.



(a) Average optimization ratio



(b) Maximum optimization ratio

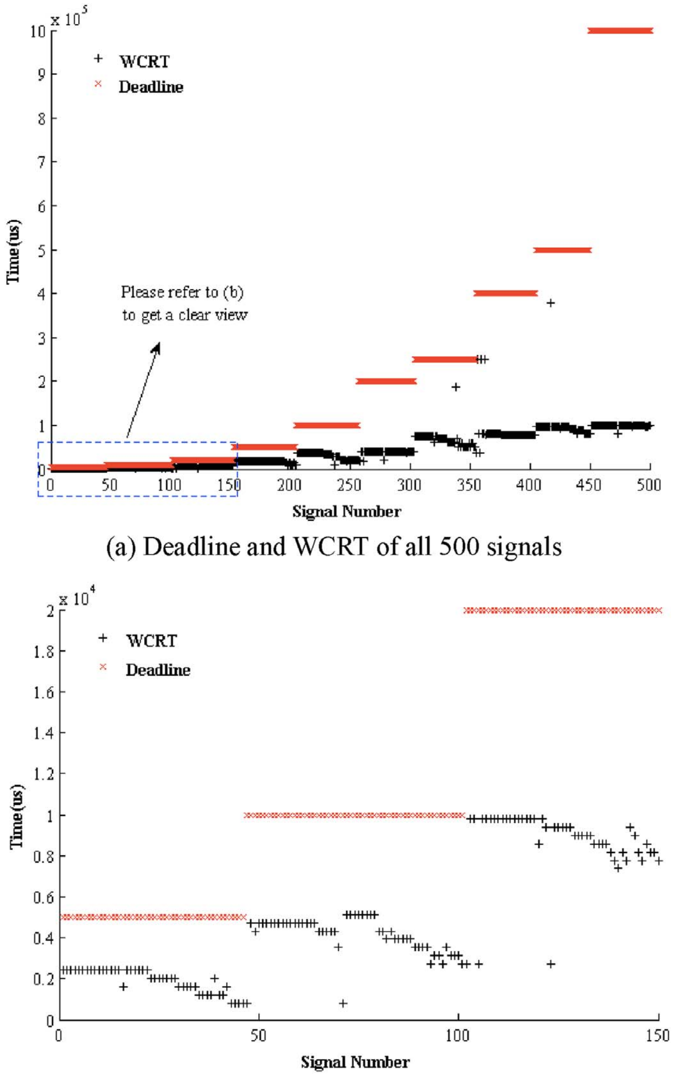
Fig. 8. Optimization ratio of bandwidth utilization for different bandwidth configurations.

signal sets is generated by NETCARBENCH 3.4 [24], where the distributions of period and size are configured as shown in Fig. 5. We generate 50 signal sets for each configuration (number of signals), and all the following experimental results are the average value of that of the 50 signal sets.

6.1. Comparison with existing design methods proposed for CAN and CAN FD

6.1.1. Signal sets with different number of signals

In the first experiments, the number of signals is configured from 60 to 340 for each signal set. The data phase bit-rate is 2 Mbps, and the



(a) Deadline and WCRT of all 500 signals

(b) Deadline and WCRT of the first 150 signals

Fig. 9. Schedulability analysis result of packing result.

arbitration phase bit-rate is 500 Kbps. We compare our method (Xie) with the four other existing methods, namely (Bor) [9], (Pol) [5], (Sak) [7], and (Uru) [10]. The execution time of the above five methods is roughly 60000 ms (Xie), 5000 ms (Bor), 1000 ms (Pol), 1000 ms (Sal) and 2000 ms (Uru), respectively. As indicated in Algorithm 2, we employed ILP to find the optimal packing result several times. Thus, the execution time of our method is significantly larger than that of other heuristic algorithms. To illustrate the necessity of the proposed design method, we also compare it with the no-pack method (each signal is a

Table 2
The first example signal set.

Signal	Period(μs)	Size(bit)
s_1	1000	30
s_2	2000	2
s_3	8000	2
s_4	8000	22

Table 3
The second example signal set.

Signal	Period(μs)	Size(byte)
s_1	1000	4
s_2	2000	1
s_3	8000	1
s_4	8000	3

Table 4
DP table when $AP = 1$ and the packing result is $m = \{s_3\}$.

	0	1	2	3	4	5	6	7	8	9
4	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0
3	∞, ∞, 0	**0.00988, 1, 8000**	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0
2	∞, ∞, 0	0.0395, 1, 2000	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0
1	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0
0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0

message). In the three experiments, the obtained largest bandwidth utilization is only approximately 70% except the no-pack method, and the packed message sets are all schedulable by using the priority assignment method of [19]. Thus, the minimization of the bandwidth utilization is considered only, and the schedulability in Section 6.2 is disregarded. The details of the comparison results are shown in Tables 6 and 7 and Figs. 6 and 7.

Tables 6 and 7 indicate the average bandwidth utilization of the packed message set for the first and the second kind of signal sets when the number of signals is configured from 60 to 340, respectively. We can find that the proposed method is the most bandwidth efficient, followed by Uru, Sak, Pol, and Bor.

Figs 6 and 7 show the average and the maximum optimization ratio of the bandwidth utilization of the proposed method by comparing it with the other methods. The full lines represent the maximum optimization ratio. For example, Xie-Bor-Max indicates the maximal optimization ratio of the proposed method by comparing it with [9]. The dash lines represent the average optimization ratio. For example, Xie-Bor-Avg indicates the average optimization ratio of the proposed method by comparing it with [9].

When the proposed method is compared with [7,5,10], the average and the maximal optimization ratio are about 5%/6%/2%, and 10%/12%/5%, respectively. When the proposed method is compared with [9], the average and the maximal optimization ratio vary from 6.32% to 22.25%, and from 8.75% to 34.87%, respectively. [5,7,9] input one signal for packing each time, so it easily leads the packing result falling into the local optimum. But our method input a signal cluster for packing each time, it gets more bandwidth-efficient result. We can also find that there is a relative small optimization ratio when comparing our method with [10], as these two methods takes the same step by clustering signals with equal period, and this step is key to the final design result.

6.1.2. CAN FD with different data phase bit-rates

CAN FD can support a data phase bit-rate of up to 10 Mbps; thus, another experiment is conducted to further verify the effectiveness of the proposed method by using the first kind of signal sets, where the

Table 5
DP table when $AP = 9$ and the packing result is $m=\{s_3, s_4\}$.

	0	1	2	3	4	5	6	7	8	9
4	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	0.01163, 3, 8000	**0.0115, 4, 8000**	0.0445, 5, 2000	0.0445, 5, 2000	0.0445, 5, 2000	0.0445, 5, 2000	0.0445, 5, 2000
3	∞, ∞, 0	0.0119, 1, 8000	0.047, 2, 2000	0.047, 2, 2000	0.047, 2, 2000	0.047, 2, 2000	0.09, 6, 1000	0.09, 6, 1000	0.09, 6, 1000	0.09, 6, 1000
2	∞, ∞, 0	0.0475, 1, 2000	0.0475, 1, 2000	0.0475, 1, 2000	0.0475, 1, 2000	0.091, 5, 1000	0.091, 5, 1000	0.091, 5, 1000	0.091, 5, 1000	0.091, 5, 1000
1	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	0.092, 4, 1000	0.092, 4, 1000	0.092, 4, 1000	0.092, 4, 1000	0.092, 4, 1000	0.092, 4, 1000
0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0

Table 6
Average bandwidth utilization of packed message set based on the first kind of signal sets (%).

Num of Sig								
Methods	60	100	140	180	220	260	300	340
Xie	12.18	19.86	26.91	33.75	42.97	53.48	57.35	67.64
Bor[9]	15.67	22.61	30.8	37.15	46.87	59.38	62.21	72.85
Sak[7]	12.72	20.9	28.3	35.33	44.99	56.05	59.73	70.64
Pol[5]	12.8	21.12	28.65	35.87	46.09	57.33	60.7	72.24
Uru[10]	12.58	20.29	27.37	34.26	43.58	54.35	58.19	68.53
No Pack	26.07	46.44	62.74	83.08	103.25	130.02	141.49	166.51

Table 7
Average bandwidth utilization of packed message set based on the second kind of signal sets(%).

Num of Sig								
Methods	60	100	140	180	220	260	300	340
Xie	11.47	18.44	24.63	32.03	37.09	44.67	48.95	58.89
Bor[9]	14.14	20.4	27.63	35.9	40.17	48.44	52.8	62.86
Sak[7]	12.16	19.28	25.79	33.21	38.74	46.37	50.74	61.39
Pol[5]	12.22	19.39	26.12	33.82	39.47	47.08	51.38	62.63
Uru[10]	11.77	18.74	25.04	32.44	37.54	45.26	49.62	59.73
No Pack	27.92	48.6	66.44	87.96	104.24	123.52	138.39	168.52

Table 8
Average bandwidth utilization for different bandwidth configurations.

Bit-Rates					
Methods	2	4	6	8	10
Xie	57.62	33.74	25.75	21.73	19.29
Bor[9]	62.74	38.44	30.35	26.29	23.86
Sak[7]	60.46	36.0	27.44	23.18	20.59
Pol[5]	61.68	36.32	27.84	23.4	23.27
Uru[10]	57.86	34.04	25.98	21.94	19.48

Algorithm 1
Signal ordering and clustering.

```
Input: signal set $S$
Output: signal clusters with $S = SC_1 \cup SC_2 \cup \ldots \cup SC_N$
1  Sort signals in order of increasing period with Merge Sort algorithm;
2  $SC\_Num=0$; //number of signal clusters
3  $Cur\_Period=0$; //period of the included signals in current signal cluster;
4  for $i=1$: $|S|$ do
5    if $t_i \neq Cur\_Period$ then
6      $SC\_Num++$;
7      $Cur\_Period= t_i$;
8    end if
9    add $s_i$ into $SC_{SC\_Num}$;
10 end for
11 return the obtained signal clusters;
```

Algorithm 2
Packing of the signal clusters.

```
Input: signal clusters from Algorithm 1
Output: message set $M$ // $M$ is initially set to{}
1  for $k=1$:$SC\_Num$ do //pack the signal clusters
2    if $k==1$ then
3      while $SC_k \neq$ null do //pack all signals of $SC_k$ into null messages
4        ILP_Pack($SC_k$, $m_{null}$, 0); //pack $SC_k$ into a null message with ILP formulation
5        add the packed message into $M$ and calculate its $PLSB$ and $MLSB$;
6        delete the packed signals from $SC_k$;
7      end while
8    else
9      while $SC_k \neq$ null do
10       if the sum($PLSB$, $MLSB$) of the messages in $M$ can accommodate $SC_k$ then
11         Bandwidth_Utilization_Analysis($M$, $SC_k$);
12         if Pack $SC_k$ into null messages is more bandwidth efficient then
13           ILP_Pack($SC_k$, $m_{null}$, 0); //pack $SC_k$ into a null message with ILP formulation
14           add the packed message into $M$ and calculate its $PLSB$ and $MLSB$;
15           delete the packed signals from $SC_k$;
16         else
17           while $SC_k \neq$ null do //pack $SC_k$ into $PLSB$ and $MLSB$ of existing messages
18             ILP_Pack($SC_k$, $m_{exist}$, 1); //pack $SC_k$ with ILP formulation
19             delete the packed signals from $SC_k$;
20             update the $PLSB$ and $MLSB$ of the enlarged messages;
21           end while
22           break;
23         end if
24       else
25         for $j=1$: $|M|$ do //pack $SC_k$ into $PLSB$ of existing messages
26           if $PLSB_j \neq 0$ do
27             ILP_Pack($SC_k$, $m_{exist}$, 2); //pack $SC_k$ with ILP formulation
28             delete the packed signals from $SC_k$ and update $PLSB_j$;
29           end if
30         end for
31         if $SC_k \neq$ null do //pack $SC_k$ into a null message
32           ILP_Pack($SC_k$, $m_{null}$, 0); //pack $SC_k$ with ILP formulation
33           add the packed message into $M$ and calculate its $PLSB$ and $MLSB$;
34           delete the packed signals from $SC_k$;
35         end if
36       end if
37     end while
38   end if
39 end for
40 return $M$;
```

Algorithm 3
ILP formulation of the signal packing problem.

```
Input: signal cluster $SC_k$, message $m_j$ and $obj\_size_j$
Output: the signal subset of $SC_k$ that can be packed into $m_j$
Constraints:
    $\forall\, s_i \in SC_k,\ m_j,\ \sum_i \chi_{i,j} \times z_i \le objsize_j$
Objective:
    $\forall\, s_i \in SC_k,\ m_j,\ sumsize_j = \sum_i \chi_{i,j} \times z_i$
maximize: $sum\_size_j$
```

data phase bit-rate of CAN FD is configured from 2 Mbps to 10 Mbps, respectively. The number of signals is 300 for the employed signal sets. The experimental results are shown in Table 8 and Fig. 8. We can see that as the data phase bit-rate increases, the bandwidth utilization decreases for all the five methods. However, the maximum and average optimization ratio of bandwidth utilization of the proposed method increases.

6.2. Schedulability analysis of the packing result

The proposed method considers the schedulability of each signal. Therefore, the third experiment is conducted to verify whether the timing constraint on each signal is met or not. We utilize the first kind of signal sets for this experiment, the number of signals is 500, and the bandwidth utilization of the packed message set is 91.61%. The packed message set has 62 messages, and 7 messages contain signals with different periods. The deadline and WCRT of each signal are shown in Fig. 9, where the signals are ordered with increasing period. Although the number of signal in the utilized signal sets is as large as 500 and the bandwidth utilization of the packed message set is close to 100%, the obtained message set is still schedulable.

7. Conclusion

An efficient design method for CAN FD is proposed to improve the utilization of its high bandwidth. The signals are first clustered and packed into messages to minimize bandwidth utilization. The schedulability of the packed message set and the original signal set is verified. The experimental results demonstrated that our method is the most bandwidth efficient while conforming to the signal level timing constraint at the same time. By comparing the proposed approach with other related works, the obtained average and maximum optimization ratio of bandwidth utilization of the proposed method varies from 1.2% to 22.27%, and from 2.6% to 34.87%, respectively. Although the proposed approach takes a longer time than the other methods, it is acceptable because it is meant to be used in design time. Cyber-security is a new design dimension that needs to be considered for in-vehicle networks. We plan to conduct a study on an efficient design space-searching algorithm for the security-aware design of CAN FD.

Acknowledgments

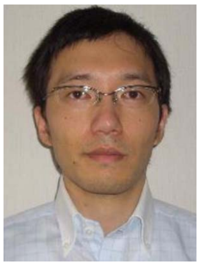
This project is supported by National Natural Science Foundation of China (61502405, 61772195, 61702172), Fujian Educational Bureau (JA15368), the CCF-Venustech Open Research Fund under Grant No. CCF-VenustechRP2017012, the Priority Academic Program Development of Jiangsu Higher Education Institutions, Jiangsu Collaborative Innovation Center on Atmospheric Environment and Equipment Technology, and the Open Research Fund of Hunan Provincial Key Laboratory of Network Investigational Technology (2016WLZC020), the CERNET Innovation Project (NGII20161003).

Reference

[1] S. Tuohy, M. Glavin, C. Hughes, et al., Intra-vehicle networks: a review, IEEE Trans. Intell. Transportation Syst. 16 (2) (2015) 534–545.
[2] Robert Bosch GmbH, CAN with flexible data-rate, White Paper, (2011) Version 1.1.
[3] F. Hartwich, CAN with flexible data-rate, Proceedings of International CAN Conference, 14 2012, pp. 1–14.9.
[4] CAN FD approved. http://can-newsletter.org/engineering/engineering-miscellaneous/ 150630_ can-fd-approved_approved_iso-11898-1_cia/, 2015-06-30.
[5] F. Polzlbauer, I. Bate, E. Brenner, Optimized frame packing for embedded systems, IEEE Embedded Syst. Lett. 4 (3) (2012) 65–68.
[6] K. Sandstrom, C. Norstrom, M. Ahlmark, Frame packing in real-time communication, International Conference on Real-Time Computing Systems and Applications, 2000, pp. 399–403.
[7] R. Saket, N. Navet, Frame packing algorithms for automotive applications, J. Embedded Comput. 2 (1) (2006) 93–102.
[8] F. Polzlbauer, I. Bate, E. Brenner, On extensible networks for embedded systems, Proceedings of the Annual International Conference and Workshop on the Engineering of Computer Based Systems, 2013, pp. 69–77.
[9] U.D. Bordoloi, S. Samii, The frame packing problem for CAN-FD, Proceedings of the IEEE Real-Time Systems Symposium, 2014, pp. 284–293.
[10] G. Urul, A Frame Packing Method to Improve the Schedulability on CAN and CAN FD[Master Thesis], The Middle East Technical University, Ankala, Turkish, 2015.
[11] P. Pop, P. Eles, Z. Peng, Schedulability-driven frame packing for multicluster distributed embedded systems, ACM Trans. Embedded Comput. Syst. 4 (1) (2005) 112–140.
[12] W. Zheng, Q. Zhu, et al., Definition of task allocation and priority assignment in hard real-time distributed systems, Proceedings of IEEE International Real-Time Systems Symposium, 2007, pp. 161–170.
[13] Q. Zhu, Y. Yang, et al., Optimizing extensibility in hard real-time distributed systems, Proceedings of IEEE Real-Time and Embedded Technology and Applications Symposium, 2008, pp. 275–284.
[14] C.W. Lin, Q. Zhu, et al., Security-aware mapping for CAN-based real-time distributed automotive systems, Proceedings of The IEEE/ACM International Conference on Computer-Aided Design, 2013, pp. 115–121.
[15] Y. Xie, L.J. Liu, et al., Security-aware signal packing algorithm for CAN-based automotive cyber-physical systems, IEEE/CAA J. Automatia Sinica 2 (4) (2015) 248–257.
[16] K.W. Tindell, A. Burns, Guaranteeing message latencies on controller area network, Proceedings of International CAN Conference, 1994, pp. 1–11.
[17] K.W. Tindell, H. Hansson, A.J. Wellings, Analysing real-time communications: controller area network(CAN), Proceedings of IEEE International Real-Time Systems Symposium, 1994, pp. 259–263.
[18] R. Davis, A. Burns, et al., Controller area network(CAN) schedulability analysis: refuted, revisited and revised, J. Real-Time Syst. 35 (3) (2007) 239–272.
[19] N.C. Audsley, On priority assignment in fixed-priority scheduling, Inf. Process. Lett. 79 (1) (2001) 39–44.
[20] R.I. Davis, A. Burns, Robust priority assignment for messages on controller area network(CAN), Real-Time Syst. 41 (2) (2009) 152–180.
[21] K.W. Schmidt, Robust priority assignment for extending existing controller area network applications, IEEE Trans. Ind. Inf. 10 (1) (2013) 578–585.
[22] R.I. Davis, A. Burns, et al., On priority assignment for controller area network when some message identifiers are fixed, Proceedings of International Conference on Real-Time and Network Systems, 2015, pp. 279–288.
[23] T.H. Cormen, C.E. Leiserson, et al., Introduction to Algorithms, third ed., MIT Press, 2009.
[24] NETCARBECH 3.4. http://www.netcarbench.org/.

Yong Xie: received his Ph.D. degree in Computer Science and Technology from Hunan University, China in 2013, was a visiting Ph.D. student of Graduate School of Information Science, Nagoya University from 2011 to 2012. He is currently a postdoctoral researcher of National Laboratory for Parallel and Distributed Processing, National University of Defensive Technology, and an associate professor of Xiamen University of Technology. His research interests include in-vehicle networks and real-time embedded systems. He is a member of IEEE and ACM.

Gang Zeng: Gang Zeng is an associate professor 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 and in-vehicle networks. He is a member of IEEE and IPSJ.



Kurachi Ryo: received his bachelor and master degrees from Tokyo University of Science in 2000 and 2007, and his Ph.D. from Nagoya University in 2012. From 2000 to 2006 he was working for AISIN AWCO., LTD. He is currently a visiting associate professor of the Center for Embedded Systems(NCES), Nagoya University. His research interests include in-vehicle networks and real-time scheduling theory.

Guoqi Xie: received his Ph.D. degree in computer science from Hunan University, China, in 2014. He was a postdoctoral researcher at Nagoya University, Japan, from 2014 to 2015. He is currently a postdoctoral researcher at Hunan University, China, since 2015. His major interests include embedded systems, distributed systems, in-vehicle networks, real-time systems, and cyber-physical systems.

Yong Dou: received his B.S., M.S., and Ph.D. degrees from National University of Defense Technology in 1989, 1992 and 1995. His research interests include high performance computer architecture, high performance embedded microprocessor, reconfigurable computing, machine learning, and bioinformatics. He is a member of IEEE and ACM.

Zhili Zhou: received his BS degree at Communication Engineering from Hubei University in 2007, and his MS and PhD degrees in Computer Application at the School of Information Science and Engineering from Hunan University, in 2010 and 2014, respectively. He is an assistant professor at Nanjing University of Information Science and Technology. His current research interests include near-duplicate image/video detection, image/video copy detection, coverless information hiding, digital forensics, and image processing.