Optimal Priority Assignment for Multiple CAN/CAN-FD Buses with a Central Gateway

Taeju Park, Jiarui Lyu, and Kang G. Shin
Department of Electrical Engineering and Computer Science
The University of Michigan – Ann Arbor
Email: {taeju, jiarui1, kgshin}@umich.edu

***Abstract*—Automakers keep introducing new functions to vehicles to entice customers, thus increasing the size/number of in-vehicle buses. As a result, adopting multi-buses with a central gateway is becoming a norm in current and future vehicles. Since adding buses increases production cost, knowing whether or not to add a bus at design time to meet the given requirements is essential to design a cost-optimized in-vehicle network. However, due to the lack of an optimal priority-assignment algorithm for multi-bus systems, it is difficult to determine whether additional buses are needed. To address this difficulty, we propose an optimal priority-assignment algorithm, called OPMB, for multiple CAN/CAN-FD buses with a central gateway. OPMB builds on backtracking, and is thus of exponential time complexity. To reduce the execution time effectively for industry-size problems, we identify several theory-proven search-space reduction conditions. Our in-depth simulation has demonstrated that OPMB outperforms the state-of-art priority-assignment algorithms for multi-bus systems, and is suitable for high-speed systems which represent future automotive systems. Also, OPMB is shown to be feasible for most realistic automotive message sets.**

I. Introduction

Automakers keep adding new functions to their products to attract more customers. Since such newly-introduced functions usually require communication with other electronic control units (ECUs) to acquire & deliver sensor (e.g., speedometer, radar, etc.) data, the amount of in-vehicle network traffic keeps rising. Due to this increase of in-vehicle traffic, the controller area network (CAN) — *de facto* standard of the in-vehicle network which supports up to 1Mbps — reaches its bandwidth limit. Thus, the automakers are adding more CAN buses and connecting them via a *central gateway* to handle the increasing vehicle data traffic. In other words, a system of multiple buses connected via a central gateway has become a norm in new vehicles. Recently, automakers have also begun replacing CAN with a higher bandwidth protocol, Controller Area Network with Flexible Data rate (CAN-FD), which can support up to 12Mbps. However, since CAN has enough bandwidth and speed to support certain domains like powertrain and is also cheaper than CAN-FD, both CAN and CAN-FD are expected to coexist in the foreseeable future.

Automakers are very conscious of production cost, and hence want to design a cost-minimized in-vehicle network. It is, therefore, important to know if a designed system satisfies the requirements of given in-vehicle messages at design time. If the system cannot meet the requirements, the system designer must modify the designed system by either adding more resources (e.g., more CAN/CAN-FD buses) or optimizing the system further. Thus, it is necessary to have an "optimal" priority assignment algorithm[1] available at design time. The optimal algorithm can then be used to determine the (non)existence of a schedulable priority assignment for the given set of messages on the designed system.

The priority-assignment problem has been studied by many researchers for decades. Of them, Audsley's optimal priority assignment (AOPA) [18] is proven optimal for a *single* CAN/CAN-FD bus. Even though the central-gateway-based architecture is commonly used in modern vehicles, to the best of our knowledge, Joshi *et al.* [11] are the first and the only one with focus on priority assignment for multi-buses with a central gateway. Their algorithm is claimed to be optimal for the multi-bus system under the assumption that central gateway cannot change priority of in-coming network traffic. However, the central gateway for automotive (e.g., AUTOSAR gateway) can easily change the priority of in-coming network messages with very little cost, and hence there is no reason to restrict priority changes at central gateway in practice. Also, to the best of our knowledge, there is no optimal algorithm proposed thus far for the multi-bus system where the central gateway can change the priority of in-coming network messages. Thus, we need an optimal priority assignment algorithm for the system.

To meet this need, we propose an **O**ptimal **P**riority-assignment algorithm for **M**ultiple CAN/CAN-FD **B**uses with a central gateway (**OPMB**). It builds on backtracking (brute-force search with theoretically-proven pruning) to assign priorities to given messages. In particular, OPMB can tell the system designers the (non)existence of priority assignments which satisfy the timing requirements of the given set of messages in a networked system. However, the brute-force search incurs exponential time complexity, making it essential to prune unnecessary searches. We have identified several ways of pruning unnecessary searches and proved that the identified pruning does not affect the discovery of a schedulable priority assignment. We have also conducted extensive simulation by generating realistic in-vehicle messages. Our simulation results show OPMB is able to determine whether schedulable priority assignment exists or not for 96.9% of realistic automotive

[1]If the optimal priority-assignment algorithm cannot find a schedulable priority assignment, no other assignment algorithm can find a schedulable priority assignment either.

message-sets *within 1 second*. Also, the results show that OPMB outperforms the state-of-art priority assignment algorithms in terms of *schedulability coverage*.[2]

This paper makes the following main contributions:

- A counter-example showing that global priority assignment cannot be optimal for a multi-bus system where the central gateway can alter the priority of in-coming network messages.
- Development of an optimal priority-assignment algorithm, OPMB, for a multiple CAN/CAN-FD bus system with a central gateway;
- Demonstration of utility of OPMB for industry-size problems and its superiority to existing priority-assignment algorithms for the system consisting of multiple CAN/CAN-FD buses with a central gateway.

The rest of paper is organized as follows. We discuss the related work in Section 2. We provide the primers of CAN and CAN-FD in Section 3, and describe the system model in Section 4. We prove that global priority assignment algorithm cannot be optimal in Section 5, and state the priority-assignment problem in Section 6. In Section 7, we present the new optimal priority-assignment algorithm, OPMB, for multiple CAN/CAN-FD buses with a central gateway. Section 8 evaluates the utility of OPMB by measuring its execution time and performance in comparison with the existing priority-assignment algorithms. We discuss the limitation of OPMB in Section 9 and conclude the paper in Section 10.

II. Related Work

A. Priority assignment for CAN/CAN-FD

Since priority assignment to CAN/CAN-FD messages greatly affects the schedulability of a given set of messages, there have been various priority-assignment algorithms proposed for CAN/CAN-FD buses. The most representative of them is Audsley's optimal priority assignment (AOPA) [1], which is proven optimal by Davis *et al.* [18] for a *single* CAN/CAN-FD bus without priority inversion, but priority inversions can occur in practice [6, 12, 13].

The variants of AOPA have also been proposed to address practical problems. For example, Davis *et al.* [7] proposed a robust priority assignment by maximizing the number of successive tolerable transmission errors without any timing violation, in order to account for transmission errors that may happen in practice. Schmidt *et al.* [19] considered backward compatibility in a priority assignment to reflect a condition in which some messages have fixed IDs. Additionally, the limitation of their approach due to the insufficient gap between fixed IDs is addressed in [8]. Joshi *et al.* [11] proposed an algorithm for multiple CAN-FD buses with a central gateway. Even though their algorithm is claimed to be optimal for multi-bus systems, we find that claim does not hold when buses have different link speeds.

[2]Defined as the ratio of the number of schedulable cases (by using a priority assignment algorithm) to the number of tested cases.

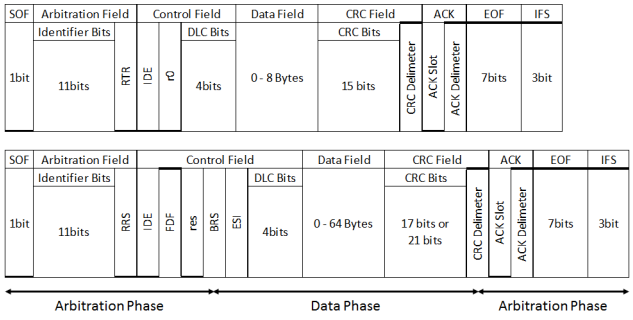

Fig. 1. CAN frame (Up) and CAN-FD frame (Down)

There are also different types of ID/priority assignment algorithms for CAN/CAN-FD messages. Pölzlbauer *et al.* [17] considered the extensibility problem of ID assignment for CAN. Park *et al.* [15] proposed a message priority-assignment algorithm for a bus shared by both CAN nodes and CAN-FD nodes where changing the operation mode of CAN controller is required. The algorithm is designed based on NP-EDF [10] to minimize the number of operation-mode changes.

B. Priority assignment for distributed real-time system

Multiple buses with a central gateway can be regarded as a distributed real-time system since we need to schedule each message on two buses to meet its end-to-end deadline. The task priority assignment in distributed real-time systems has been studied extensively. This problem is NP-hard [22], and hence various heuristic algorithms have been proposed. Garcia *et al.* [9] proposed a heuristically optimized priority assignment (HOPA) by leveraging those design parameters affecting the worst-case response time. Azketa *et al.* [4] proposed use of the genetic algorithm to find a schedulable priority assignment. Yoon *et al.* [23] proposed zero slack priority assignment (ZSPA), which decomposes an end-to-end deadline into local per-task deadlines and assigns priorities to the sub-tasks using AOPA. If the worst-case response time of a sub-task after assigning a priority is smaller than its local deadline, the remaining slack is used for other sub-tasks. Thus, initially-determined local deadlines are automatically adjusted in the priority assignment.

III. CAN and CAN-FD Primers

We briefly introduce necessary basics of CAN and CAN-FD including frame format, bit rate change, message arbitration, and the analysis of a CAN message's worst-case latency.

A. Frame Format

As shown in Fig. 1, CAN and CAN-FD frames consist of start-of-frame (SOF), arbitration, control, data, cyclic redundancy check (CRC), acknowledgment (ACK), and end-of-frame (EOF) fields. The SOF, ACK and EOF fields of a CAN-FD frame are the same as those of a CAN frame. However, there are several differences between CAN and CAN-FD frames in the arbitration, control, data, and CRC

TABLE I
SUPPORTED PAYLOAD SIZE

Data Frame Type	DLC	Payload Size	CRC Bits
CAN & CAN-FD	0000	0	CAN:15 CAN-FD:17
	0001	1	
	0010	2	
	0011	3	
	0100	4	
	0101	5	
	0110	6	
	0111	7	
CAN	1xxx	8	
CAN-FD	1000	8	17
	1001	12	
	1010	16	
	1011	20	21
	1100	24	
	1101	32	
	1110	48	
	1111	64	

fields. Especially, two special bits are added in the control field of CAN-FD for boosting the transmission speed.

- Flexible Data-rate Format (FDF) bit indicates whether the frame is encoded as CAN frame (dominant bit) or CAN-FD frame (recessive bit).
- Bit Rate Switch (BRS) bit indicates whether the bit time is changed in data phase (recessive bit) or not (dominant bit).

Table I shows CAN and CAN-FD payload sizes. Since CAN only supports at most 8 bytes data per frame, this small payload size has been the roadblock in implementing several functions, such as fast flashing of an updated firmware. To solve these problems, CAN-FD supports up to 64 bytes data per frame. However, since only 4 bits are used for DLC, it is not enough to specify the payload size of 0–64. Thus, the discretized payload size is supported as shown in Table I. Also, the increased payload size requires more redundant bits to check the correctness in transmitted data, thus increasing the number of CRC bits along with the payload size.

B. Switching bit rate

Unlike CAN, CAN-FD is separated into two phases, arbitration and data phases, as shown in Fig. 1. The interval between BRS bit and CRC delimiter bit is defined as the data phase. The other intervals are defined as the arbitration phase.

The purposes of this separation are to support the CAN-FD's improved transmission rate and preserve the key features of CAN, such as the non-destructive arbitration mechanism. Thus, CAN-FD defines two bit times: *nominal bit time* (t_{nom}) and *data bit time* (t_{data}). These two bit times are configured with the consideration of properties of a CAN-FD network (e.g., the number of ECUs, the length of wire, limitations of CAN-FD transceivers, etc.) when a CAN-FD controller is initialized. Note that t_{data} has to be smaller than or equal to t_{nom}. The bit time for the arbitration phase is always configured to t_{nom}. However, the bit time for the data phase is determined by the BRS bit value. If the BRS bit of a CAN-FD frame is 1 (recessive bit), then the bit time is switched from t_{nom} to t_{data} in order to bump up the transmission rate. Otherwise, t_{nom} holds for the data phase.

C. Message scheduling on CAN/CAN-FD bus

CAN/CAN-FD schedules messages based on the value of 11-bits (or 29-bits) identifier (ID) field of messages in a decentralized way. When multiple ECUs transmit CAN messages at the same time, the message with the lowest ID value is selected to be transmitted under the CAN protocol. ECUs take or filter out the transmitted message based on the bit-pattern in ID field using a hardware filter. If a message loses the arbitration, it will contend again after the transmission of the currently selected message. So, by assigning IDs properly, the system designer can control the order of transmitting messages. That is, priority assignment to CAN messages is equivalent to ID assignments.

D. Timing Analysis of CAN/CAN-FD

Applications that exchange messages on a CAN/CAN-FD bus are often time-critical (e.g., engine control commands). Thus, analyzing the worst-case response time (WCRT) of CAN/CAN-FD messages and proving their WCRT to be less than their deadline is essential for time-critical applications. We will henceforth use the WCRT and the worst-case delay interchangeably.

Davis *et al.* [18] proposed a way of analyzing the WCRT of CAN messages on a single bus. They decompose the WCRT (R_i) of a CAN message (m_i) into three parts: (1) release jitter (J_i), (2) queuing delay of the q-th instance of CAN message at a transmission buffer in CAN controller ($w_i(q)$), and (3) transmission time of the message on the bus (C_i). Since the worst-case release jitter and the transmission time on a bus are constant, they focused on analysis of the queuing delay at the transmission buffer.

To derive the worst-case queuing delay of a message, they analyzed critical instant and busy period for every instance of a message. They recursively derived the worst-case queuing delay of a message as:

$$w_i^{n+1}(q) = B_i + qC_i + \sum_{\forall k \in hp(i)} \left\lceil \frac{w_i^n(q) + J_k + \tau}{p_k} \right\rceil C_k. \quad (1)$$

Note that if $w_i^{n+1}(q) = w_i^n(q)$ or $w_i^n(q)$ is larger or equal to deadline of m_i, the recursive computation terminates. They selected the maximum of the response times for calculated instances as the message's WCRT:

$$R_i(q) = J_i + w_i(q) - qp_i + C_i \quad (2)$$

$$R_i = \max_{q=0,\ldots,q^{max}} R_i(q). \quad (3)$$

where p_i is the message's period, B_i is the blocking time by a lower-priority message on the CAN bus, *hp*(i) is a set of messages whose priorities are higher than m_i, and τ is a bit time. Since the q-th instance of the message is released at qp_i, qp_i is subtracted from the completion time of the q-th instance

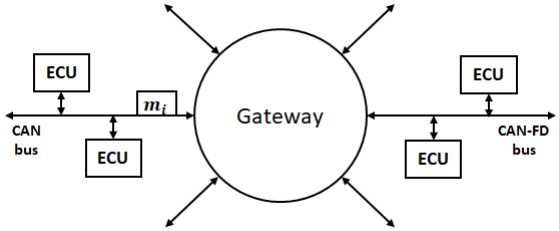

Fig. 2. System model of multiple CAN/CAN-FD buses with a central gateway

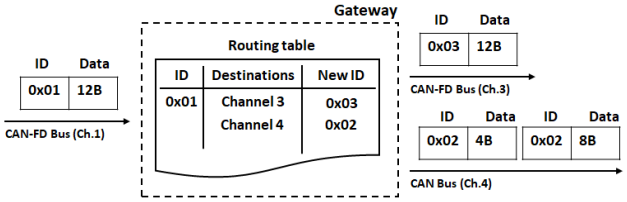

Fig. 3. Message routing in the gateway based on the embedded routing table

to calculate the response time as in Eq. (3). The q^{max} can be derived by computing the longest busy period (LBP_i) as:

$$LBP_i = B_i + \sum_{\forall k \in hp(i) \cup \{i\}} \left\lceil \frac{LBP_i + J_k + \tau}{p_k} \right\rceil C_k. \quad (4)$$

$$q^{max} = \left\lceil \frac{LBP_i + J_i}{p_i} \right\rceil - 1 \quad (5)$$

Message scheduling and queuing for CAN-FD is the same as those for CAN, so we can apply the above equations directly to compute the WCRT of a CAN-FD message. The only difference between them is the computation of the transmission time (C_i) since the bit-rate can be switched during a transmission under CAN-FD.

Although there are variations [12, 13, 5, 3, 16] of the above analysis to consider various practical issues that could affect the WCRT of a CAN message, we will use the basic analysis in this paper.

IV. System Model

Fig. 2 illustrates the system under consideration which is composed of multiple CAN/CAN-FD buses, ECUs, and a central gateway. This central gateway-based networked system is commonly used in modern vehicles (e.g., Volkswagen Atlas 2018). Several ECUs and the gateway are connected to a bus, and they use the shared medium to transmit messages to other ECUs or the gateway on the bus.

A. Bus and message models

If every ECU connected to a shared bus has the ability to receive a CAN-FD data frame using its CAN-FD controller/transceiver, we call the shared bus *CAN-FD bus*. Otherwise, we call the shared bus *CAN bus*. We assume that only messages compatible with the CAN data frame format can be transmitted on a CAN bus. Likewise, we assume that only messages compatible with the CAN-FD data frame format can be transmitted on a CAN-FD bus. Thus, a bus (b_i) can be modeled as $b_i = \{\vec{ecu}_i, type_i, ls_i^{arb}, ls_i^{data}\}$ where

- $\vec{ecu}_i$: a set of ECUs connected to b_i;
- $type_i \in \{CAN, FD\}$: type of b_i;
- ls_i^{arb}: link speed of b_i during the arbitration phase;
- ls_i^{data}: link speed of b_i during the data phase.

ECUs generate periodic messages (m_i) which usually contain sensor data and/or control commands. We call the ECU that generates (receives) a message m_i *source* (*destination*) ECU of m_i and the bus connected to the ECU source (destination) bus. Since a sensor data can be used by multiple applications on different ECUs, a message can have more than one destination ECU. When the source ECU of a message and its destination ECU(s) are on different buses, the message is forwarded and routed to the destination ECU(s) via the central gateway. In addition, due to the characteristics of vehicle functions, the sensor data or control command carried in a message has a *valid time*, thus imposing a timing constraint on the message. For example, a braking command from an ADAS[3] application (e.g, cruise control) should be delivered to the brake module within 10ms to avoid a crash. Consequently, a message is modeled as $m_i = \{src_i, \vec{dest}_i, p_i, d_i, l_i\}$, where

- src_i: source ECU of m_i;
- $\vec{dest}_i$: a set of destination ECUs of m_i;
- p_i: period of m_i;
- d_i: relative deadline of m_i;
- l_i: payload size of m_i.

Since a message can be transmitted on multiple (source and destinations) buses, we need to treat a message on different buses as different messages so as to assign different priorities for each bus. Thus, we let $m_{i,j}$ denote m_i on bus j. In our priority-assignment algorithm, $m_{i,j}$ has a "local" deadline $d_{i,j}$, so $m_{i,j}$ is described as $\{p_{i,j}, d_{i,j}, l_{i,j}, tr_{i,j}, \chi_{i,j}\}$, where

- $p_{i,j}$: period of $m_{i,j}$, ($p_{i,j} = p_i$);
- $d_{i,j}$: relative deadline of $m_{i,j}$;
- $l_{i,j}$: payload size of m_i, ($l_{i,j} = l_i$);
- $tr_{i,j}$: transmission time of $m_{i,j}$;
- $\chi_{i,j}$: class of $m_{i,j}$ ($\chi_{i,j} \in \{NOF, SRC, DEST\}$).

We classify a message $m_{i,j}$ to be one of three types:

- $\chi_{i,j} = NOF$ if the message is not forwarded through the central gateway, i.e., m_i is only transmitted on bus j;
- $\chi_{i,j} = SRC$ if m_i is forwarded via the gateway and j is the source bus;
- $\chi_{i,j} = DEST$ if m_i is forwarded via the gateway and j is one of the destination buses.

B. Gateway model

The central gateway processes in-coming messages to be routed to their destination ECUs based on the embedded routing table in the gateway. Fig. 3 illustrates the message routing in the gateway. If the source ECU and a destination ECU are connected to the same type of buses (e.g., source and destination ECUs are connected to a CAN bus), only

[3]Advanced Driver-Assistance System

the identifier (ID) field of the in-coming message is changed and the message is forwarded to the destination ECU like in AUTOSAR [4] PDU-based routing [2].

On the other hand, if the source and destination ECUs are connected to different types of bus (e.g., the source ECU is connected to CAN bus, but the destination ECU is connected to CAN-FD bus), a frame format conversion is required. A conversion from CAN frame to CAN-FD frame is simple because only the header and tail format changes are required.

However, a conversion from CAN-FD frame to CAN frame is tricky because the maximum payload size of CAN-FD frame is much larger than that of a CAN frame. For example, a 16-byte CAN-FD frame cannot be transformed into a single CAN frame because the maximum payload size of CAN frame is 8 bytes. If a CAN-FD frame with the payload of $>$ 8 bytes needs to be forwarded to a CAN bus, the gateway must split the CAN-FD frame into multiple CAN frames. For example, if a 12-byte CAN-FD frame has to be forwarded to a CAN bus, then the CAN-FD frame will be fragmented into one 8-byte CAN frame and one 4-byte CAN frame by the gateway. The gateway then applies the same ID to the fragmented frames based on the routing table as described in Fig. 3.

Our model accounts for the conversion of a CAN-FD frame to multiple CAN frames by summing up the transmission times of the fragmented frames. For example, if 12-byte CAN-FD frame is fragmented into one 8-byte CAN frame and another 4-byte CAN frame to transmit on CAN bus j, then $tr_{i,j}$ is the sum of the transmission times of 8-byte and 4-byte CAN frames.

V. Global priority assignment vs. per-bus priority assignment for a CAN/CAN-FD multi-bus system

There are two possible ways of assigning priorities to messages for a multi-bus system: (1) global priority assignment and (2) per-bus priority assignment. Global priority assignment algorithms assign a unique priority to each message for the entire system. For example, under a global priority assignment policy, if the priority of m_i on b_1 is 1, then that of m_i on any other bus is also 1. Also, the priority of m_i should be different from the priority of m_j if $m_i \neq m_j$. The Modified Audsley's Algorithm (MAA) [11] is an example of global priority assignment. On the other hand, per-bus priority assignment algorithms assign a unique priority to each message on a bus. For example, under a per-bus priority assignment policy, any value can be the priority of m_i on b_j if the value is a unique priority for b_j. ZSPA [23] is an example of per-bus priority assignment. We compare these two priority assignments with respect to implementation and schedulability.

A. Implementation

Since the value in the identifier field represents the priority of a CAN/CAN-FD message, to implement the per-bus priority, the central gateway must be able to change the ID value of in-coming messages according to the embedded routing table. However, to implement global priority, the central gateway only needs to forward incoming messages to destination buses according to the embedded routing table. Thus, additional memory space is needed for the per-bus assignment as a penalty to store new IDs for incoming messages in the central gateway. The required additional memory space increases with the size of routing table. For example, if we assume there are 500 entries in the routing table, additional $500 \times 2Bytes$ (assuming 2Bytes used for an ID) for the new ID are required. Also, the execution time for changing ID is also the penalty of per-bus assignment. However, because changing the value of ID requires a single memory copy instruction, the execution time would be very small.

[4] AUTomotive Open System ARchitecture

B. Schedulability

Even though global priority assignment has advantage in memory usage over per-bus priority assignment, it has a disadvantage in finding schedulable priority assignment. The following example shows that global priority assignment cannot be optimal for a CAN/CAN-FD multi-bus system where the gateway can change the IDs of messages.

- $b_1 = \{\{ecu_1\}, 0, 1Mbps, 1Mbps\}$;
- $b_2 = \{\{ecu_2\}, 1, 500Kbps, 5Mbps\}$;
- $m_1 = \{ecu_1, ecu_2, 400us, 400us, 8byte\}$;
- $m_2 = \{ecu_1, ecu_2, 400us, 400us, 8byte\}$.

With a global priority assignment, there are two ways of priority assignment for the above example. The first way is that m_1 has priority 1 (higher) and m_2 has priority 2 (lower). In this case, the WCRT of m_1 on b_1 is $135\mu s$, and that of m_1 on b_2 is $86.6\mu s$ according to Eq. (3). Thus, the worst-case end-to-end (e2e) delay of m_1 ($135\mu s + 86.6\mu s = 221.6\mu s$) is smaller than its deadline ($400\mu s$). However, the WCRT of m_2 on b_1 is $270\mu s$, and that of m_2 on b_2 is $173.2\mu s$. Thus, the worst-case e2e delay of m_2 ($270\mu s + 173.2\mu s = 443.2\mu s$) exceeds its deadline ($400\mu s$).

The second way is that m_1 has priority 2 (lower), and m_2 has priority 1 (higher). However, the worst-case e2e delay of m_1 is larger than its deadline. Thus, any global priority assignment algorithms cannot find a schedulable priority assignment for this example.

However, there is a schedulable priority assignment if the same messages on different buses can have different priorities (per-bus priority assignment) as follows: (priority 1 to $m_{1,1}$), (priority 2 to $m_{2,1}$), (priority 1 to $m_{2,2}$), and (priority 2 to $m_{1,2}$). With these priority assignments, the WCRT of m_1 on b_1 is $135\mu s$, and that of m_1 on b_2 is $173.2\mu s$ according to Eq. (3). Thus, the worst-case e2e delay of m_1 ($135\mu s + 173.2\mu s = 308.2\mu s$) is less than its deadline ($400\mu s$). Also, the worst-case e2e delay of m_2 ($270\mu s + 86.6\mu s = 356.6\mu s$) is less than its deadline ($400\mu s$). Hence, global priority assignment cannot be optimal.

VI. Problem Statement

Determining whether a designed in-vehicle network can meet the requirements of a given set of messages is of great

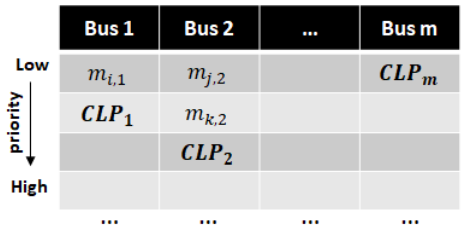

Fig. 4. Priority-assignment table

importance to minimization of the in-vehicle network cost, thus calling for an optimal priority-assignment algorithm.

To meet this need, we first want to determine whether there exists a schedulable priority assignment for a given set of messages $M = \{m_1, \ldots, m_n\}$ on a designed network of buses $B = \{b_1, \ldots, b_m\}$ such that

$$\forall i, delay^+_{i,src} + delay^+_{cgw} + delay^+_{i,dest} \leq d_i,$$

where $delay^+_{i,src}$ is the worst-case delay on the source network (the time between m_i's release at src_i and its arrival at a Rx buffer of the central gateway), $delay^+_{cgw}$ is the worst-case processing delay in the gateway (the time between m_i's arrival at the Rx buffer of the source network and its arrival at the Tx buffer of the destination network within the gateway), and $delay^+_{i,dest}$ is the worst-case delay on the destination network (time between m_i's arrival at the Tx buffer of the destination network in the gateway and its arrival at the destination ECU).

Our additional goal is to find and provide a schedulable priority assignment, if exists, for the given set M of messages on the set B of buses.

VII. OPMB

We now present an Optimal Priority assignment for Multi CAN/CAN-FD Buses (OPMB). OPMB is a backtracking-based priority-assignment algorithm. Since backtracking is basically a brute-force search with theoretically-proven pruning, it can always determine whether a solution exists or not. That is, our algorithm is intrinsically optimal priority assignment algorithm. However, without efficient pruning, it may consume a huge amount of time before it terminates. Thus, we need to overcome the key challenge of identifying theoretically-proven pruning.

A. Input parameters and return values

Before delving into OPMB, we first need to define the state of buses and the failure state, S and S_{fail}, respectively, which are used as the input parameter and the return value in OPMB.

The state S of buses consists of the states of individual buses, i.e., $S = \{S_1, \ldots, S_n\}$. The state S_j of a bus j consists of a set of assigned messages (AM_j), a set of unassigned messages (UM_j) on the bus, and the current lowest priority (CLP_j) of the bus. Thus, $S_j = \{AM_i, UM_i, CLP_i\}$. For example, if there is no assigned message for bus j, then $CLP_j = 1$ (the lowest priority) as shown in Fig. 4 (Bus m). If there is one assigned message for bus j, then $CLP_j = 2$ (the second lowest priority) as shown in Fig. 4 (Bus 1), and so on. Also, If the source or the destination of a message (m_i) is on bus j, it ($m_{i,j}$) must belong to either AM_j or UM_j.

The failure state FS consists of the index of failed bus j and the state of bus j at the time of failure $S^f_j \leftarrow S_j$, and thus $FS = \{j, S^f_j\}$.

Algorithm 1: OPMB

Input : S: the current state of buses
Output: S_{fail}: the failure state

```
1  if isSolutionFound(S) == true then
2  |  return NULL;
3  end
4  S_fail ← NULL;
5  schd ← getSchedulableAssignments(S);
6  fix ← getFixableAssignment(S);
7  if j = failureCheck(schd) then
8  |  S_fail.bus ← j;
9  |  S_fail.state ← S.S_j;
10 |  return S_fail;
11 end
12 if fix ≠ ∅ then
13 |  S' ← applyFixableAssignment(S, fix);
14 |  S_fail ← OPMB(S')
15 |  return S_fail;
16 end
17 for i ← 1 to |schd| do
18 |  correctable ← false;
19 |  if S_fail ≠ NULL then
20 |  |  if schd[selIdx].bus == S_fail.bus or
   |  |    isCrpdInUM_f(schd[selIdx], S_fail) then
21 |  |  |  correctable = true;
22 |  |  end
23 |  |  if correctable == false then
24 |  |  |  continue;
25 |  |  end
26 |  end
27 |  selIdx ← i;
28 |  S' ← applySchedulableAssignment(S, schd[i]);
29 |  S_fail ← OPMB(S');
30 |  if S_fail == NULL then
31 |  |  return NULL;
32 |  end
33 end
```

B. Initial state

In the initial state of a bus, every message belongs to the unassigned message set, i.e., $\forall i, j\ m_{i,j} \in UM_j,\ AM_j = \emptyset$. Also, CLP_j is initialized to 1 (the lowest priority for each bus) in the initial state.

Since $m_{i,j}$ needs a local deadline, we have to assign it a local deadline $d_{i,j}$. Initially, we assume that $m_{i,j}$ can fully consume the given time margin d_i for m_i. Thus, we assign the value of d_i to $d_{i,j}$ if $\chi_{i,j} = SRC$ or $\chi_{i,j} = NOF$, and we assign the value of $d_i - tr_{i,src_i}$ to $d_{i,j}$ if $\chi_{i,j} = DEST$. For example, if $d_i = 5$ and m_i is transmitted on buses 1 and 2, then $d_{i,1} = 5$ and $d_{i,2} = 5 - tr_{i,1}$. The local deadline is changed during the execution of OPMB.

After assigning the local deadline, we subtract $delay^+_{cgw}$ from $d_{i,j}$ if $\chi_{i,j} = SRC$ or $\chi_{i,j} = DEST$ to ignore

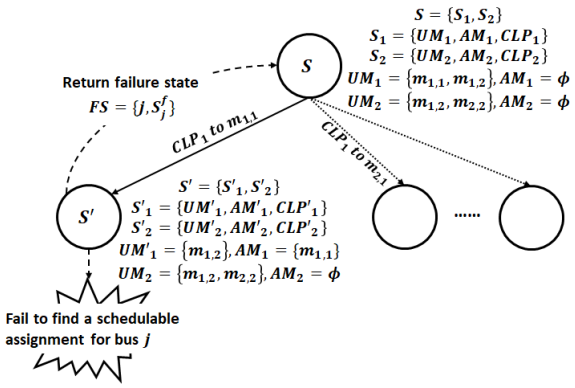

Fig. 5. Overall procedure of OPMB

the gateway processing time in the process of OPMB. For example, if $delay^+_{cgw}$ is 1, then $d_{i,1} = 4$ and $d_{i,2} = 4 - tr_{i,1}$.

C. OPMB *overall procedure*

OPMB is designed to fill the priority assignment table shown in Fig. 4. Algorithm 1 and Fig. 5 describe the overall algorithm flow of OPMB, which is implemented recursively. On each function call, OPMB assigns CLP_j to unassigned messages $m_{i,j} \in UM_j$, and calls OPMB recursively for the reduced problem (S') as illustrated in Fig. 5. If assigning priority to every message is completed successfully, OPMB returns $NULL$. To determine the assignment to apply, OPMB first finds the set of schedulable assignments and a fixable assignment for the given state S. Note that we will define and detail the schedulable and fixable assignments. Before applying either a schedulable or a fixable assignment, OPMB checks whether there is a failure condition in the given state S as stated in Lines 7-11 of Algorithm 1. If there is no schedulable assignment for bus j and $UM_j \neq \emptyset$, the *failureCheck* procedure returns the failed bus j as well as its state.

If there is a fixable assignment, then OPMB applies that assignment for the given state S. Otherwise, OPMB applies a schedulable assignment. If OPMB gets a failure return for the reduced problem (with a schedulable assignment), OPMB tries another schedulable assignment. For instance, suppose $m_{1,1}$, $m_{1,2}$, $m_{2,1}$ and $m_{2,2}$ are unassigned messages for the given state S as illustrated in Fig. 5. Also, suppose OPMB assigns priority CLP_1 to $m_{1,1}$ and calls OPMB with the reduced problem (S'), recursively. If OPMB gets a failure return for the reduced problem, OPMB tries another way of assigning priorities to unassigned messages (e.g., CLP_1 to $m_{2,1}$).

D. *Pruning unnecessary searches*

1) **Schedulable assignments:** For a given state S, if a certain assignment immediately violates the requirement, we have to discard the branch with that assignment and select a different assignment to find a solution for the given state. Thus, for the given state, OPMB discards any assignment that violates timing constraints ($delay^+_{i,j} > d_{i,j}$) as shown in Lines

Algorithm 2: getSchedulableAssignment

Input : S: the current state of buses
Output: $schd$: a set of schedulable assignment

```
1  schd ← φ
2  UM = UM_1 ∪ ... ∪ UM_m // m is the number of buses
3  for m_{i,j} ∈ UM do
4      compute delay^+_{i,j} by applying CLP_j to m_{i,j}
5      if delay^+_{i,j} ≤ d_{i,j} then
6          add (CLP_j to m_{i,j}) to schd
7      end
8  end
9  delList ← φ
10 for (CLP_j to m_{i,j}) ∈ schd do
11     for (CLP_j to m_{p,j}) ∈ schd do
12         removeFlag ← true
13         for k ← 1 to |B| do
14             if m_{i,k} exists and m_{p,k} exists then
15                 if min(d_{i,k}, d_i − delay^+_{i,j}) ≤
                     min(d_{p,k}, d_p − delay^+_{p,j}) then
16                     removeFlag ← false
17                 end
18             end
19         end
20         if removeFlag == true then
21             add (CLP_j to m_{p,j}) to delList
22         end
23     end
24 end
25 return schd − delList;
```

5-7 of Algorithm 2. Note that we compute $delay^+_{i,j}$ based on Eq. (3). OPMB only selects schedulable assignments which are defined as:

- CLP_j to $m_{i,j}$ when $delay^+_{i,j} \leq d_{i,j}$.

Suppose OPMB applies a schedulable assignment (CLP_j to $m_{i,j}$), then $delay^+_{i,j}$ is determined. So, the local deadline of $m_{i,j}$'s corresponding messages ($\forall k$ $m_{i,k}$ such that $\chi_{i,k} \neq \chi_{i,j}$) has to be re-evaluated because the amount of time can be used by $m_{i,k}$ is reduced, i.e., $d_{i,k}$ is re-evaluated after applying a schedulable assignment as:

$$d_{i,k} = \min(d_{i,k}, d_i - delay^+_{i,j}).$$

We can further prune the unnecessary searches by excluding some schedulable assignments from the set of schedulable assignments. Suppose a schedulable assignment A assigns CLP_j to the message $m_{i,j}$ and a schedulable assignment A' assigns CLP_j to the message $m_{p,j}$. Also, suppose that if $m_{i,k}$ exists, then $m_{p,k}$ also exists for every bus k. We can exclude the assignment A' from the set of schedulable assignments if $\min(d_{p,k}, d_i - delay^+_{p,j}) < \min(d_{i,k}, d_i - delay^+_{i,j})$ for every bus k. Because the corresponding messages on the other bus can have a larger local deadline by selecting A than selecting A', A is a better choice than A'. Note that large time margin is always better than small time margin to be schedulable. Thus, we exclude A' from the set of schedulable assignments as stated in Line 25 of Algorithm 2.

2) **Fixable assignments:** Suppose state S becomes S' when we select an assignment as illustrated in Fig. 6. If the

Algorithm 3: getFixableAssignment

Input : S: the current state of buses
Output: fix: a set of fixable assignments

$fix \leftarrow \phi$;
$UM = UM_1 \cup ... \cup UM_m$; // m is the number of buses
for $m_{i,j} \in UM_j$ **do**
  compute $delay^+_{i,j}$ by applying CLP_j to $m_{i,j}$;
  if ($\chi_{i,j} == NOF$ **and** $delay^+_{i,j} \leq d_{i,j}$) **or** ($\chi_{i,j} == DEST$ **and** $m_{i,src_i} \in AM_{src_i}$ **and** $delay^+_{i,j} \leq d_{i,j}$) **then**
    add (CLP_j *to* $m_{i,j}$) to fix;
    continue;
  end
  $addFlag \leftarrow true$;
  for $k \leftarrow 1$ **to** m **do**
    if $m_{i,k}$ *exists* **and** $\chi_{i,j} == SRC$ **and** $\chi_{i,k} == DEST$ **and** $delay^+_{i,j} + delay^+_{i,k} > d_i - delay^+_{cgw}$ **then**
      $addFlag \leftarrow false$;
      break;
    end
  end
  if $addFlag == true$ **then**
    for $k \leftarrow 1$ **to** $|B|$ **do**
      if $m_{i,k}$ *exists* **then**
        add (CLP_k *to* $m_{i,k}$) to fix;
      end
    end
  end
end
return fix;

non-existence of a solution for S' can guarantee the non-existence of a solution for S, we do not need to search a solution with other assignments for the given state S. Thus, the number of assignments we have to explore for the given state S becomes 1. We call such assignments *fixable-assignments* for the given state S. For example, in Fig. 5, if failure to find a solution with the assignment of CLP_1 to $m_{1,1}$ can guarantee the non-existence of a solution for the given state, we do not need to try any other assignments such as CLP_1 to $m_{2,1}$. We have identified three fixable-assignments as:

- FA1: CLP_j to $m_{i,j}$ when $\chi_{i,j} = NOF$, $delay^+_{i,j} \leq d_{i,j}$.
- FA2: CLP_j to $m_{i,j}$ when $\chi_{i,j} = DEST$, $m_{i,src_i} \in AM_{src_i}$, $delay^+_{i,j} \leq d_{i,j}$.
- FA3: CLP_j to $m_{i,j}$ and CLP_k to $m_{i,k}$ when $\chi_{i,j} = SRC$, $\chi_{i,k} = DEST$, $\forall k$ $delay^+_{i,j} + delay^+_{i,k} \leq d_i - delay^+_{cgw}$.

How to obtain these fixable assignments for a given state S is described in Algorithm 3.

Lemma 1. *Suppose the given state S is changed to S' after applying a FA1. Then, the non-existence of a solution for S' guarantees the non-existence of a solution for S.*

Proof: Since $\chi_{i,j} = NOF$, m_i is not forwarded via a central gateway. Thus, the assignment (CLP_j to $m_{i,j}$) only affects the state of bus j (S_j) from the given state S. Hence, the assignment only affects the message schedulability on bus j.

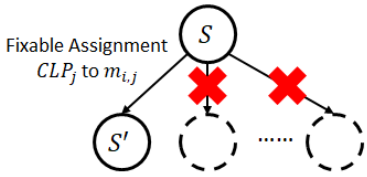

Fig. 6. If CLP_j to $m_{i,j}$ is a fixable assignment for a given state S, we do not need to move forward with other assignments from S.

Suppose there is no solution for S', but a solution exists for S. That is, every message on bus j is schedulable in state S, but the assignment makes $\exists k$ $m_{k,j} \in UM_j$ unschedulable.

When CLP_j is assigned, $delay^+_{k,j}$ of messages ($\forall k$ $m_{k,j} \in UM_j$) is not affected by the assignment. Also, the assignment does not change $d_{k,j}$. Hence, the assignment does not affect the schedulability of $\forall k$ $m_{k,j} \in UM_j$. That is, the assignment cannot make $\forall k$ $m_{k,j} \in UM_j$ unschedulable. It contradicts the supposition. Thus, if there is no solution for S', then there is no solution for S.

Lemma 2. *Suppose the given state S is changed to S' after applying a FA2. Then, the non-existence of a solution for S' guarantees the non-existence of a solution for S.*

Proof: Since $\chi_{i,j} = DEST$ and $m_{i,src_i} \in AM_{src_i}$, the assignment CLP_j to $m_{i,j}$ only changes the state of bus j (S_j) from the given state S. Thus, the assignment can only affect the message schedulability on bus j. Similarly to the proof of Lemma 1, we can show that the assignment cannot make $\forall k$ $m_{k,j} \in UM_j$ unschedulable. Thus, if there is no solution for S', then there is no solution for S.

Lemma 3. *Suppose the given state S is changed to S' after applying a FA3. Then, the non-existence of a solution for S' guarantees the non-existence of a solution for S.*

Proof: Unlike FA1 and FA2, FA3 makes multiple assignments at once, and changes the state of multiple buses (j and k) from the given state S. Thus, FA3 can affect the message schedulability on the multiple buses.

Suppose there is no solution for S', but a solution exists for S. It means that every message on the multiple buses is schedulable in the state S, but the assignments make at least one unassigned message on the buses unschedulable.

Since CLP_j is assigned to $m_{i,j}$ and CLP_k is assigned to $\forall k$ $m_{i,k}$ at once, the assignments do not change any local deadline of messages on the buses. Also, the worst-case delay of unassigned messages on the buses is not affected by the assignments. Thus, like the proofs of Lemma 1 and 2, the assignments cannot make a message unschedulable on the buses. Thus, if there is no solution for S', then there is no solution for S.

3) **Restriction from Failure State***:* Suppose OPMB performs a schedulable assignment A in the given state S, and the assignment returns a failure state. To find a solution for the state S, OPMB tries a different schedulable assignment

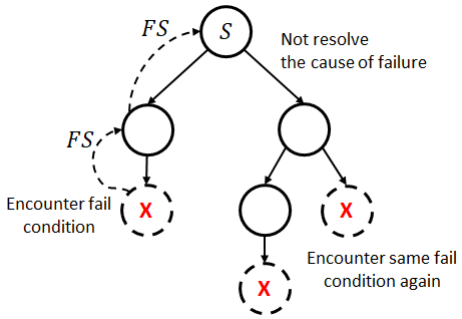

Fig. 7. Encountering failure without resolving its cause

(A') if exists. However, if A' does not resolve the cause of failure OPMB experienced with A, OPMB will encounter the same failure again as illustrated in Fig. 7. Thus, to prune the unnecessary search (A'), OPMB checks whether A' can potentially resolve the received failure state or not.

When OPMB cannot find any schedulable assignment for bus j, OPMB returns the failure state ($FS = \{j, S_j^f\}$). To resolve the failure state (there is no schedulable message in UM_j^f), at least one applied assignment related to UM_j^f has to be revoked. Thus, OPMB allows A' to be tried only when A is related to UM_j^f. In other words, OPMB allows an assignment (A') to be tried only when (1) A assigns CLP_j to a message on the failed bus j or (2) A assigns a priority to the corresponding message in UM_j^f as stated in Lines 19–26 in Algorithm 1.

Lemma 4. *Suppose* OPMB *fails to find a schedulable priority assignment for a given state S with an assignment A and get a failure state $FS = \{j, S_j^f\}$. Also, suppose A does not assign priority to a message on the failed bus j and A does not assign priority to the corresponding message in UM_j^f. Then,* OPMB *cannot find a schedulable priority assignment with any other schedulable assignment (A') for the given state S.*

Proof: Suppose OPMB can find a schedulable priority assignment with an arbitrary assignment A', i.e., every message on bus j is schedulable with A'. In other words, assignment A makes messages in UM_j^f unschedulable. However, the local deadlines of messages in UM_j^f are not affected by the assignment A because A does not assign priority to the corresponding message in UM_j^f. Also, the worst-case delays of messages in UM_j^f are not affected by the assignment A because A does not assign priority to a message on the failed bus j. Hence, A cannot make a message in UM_j^f unschedulable. It contradicts the supposition. Thus, OPMB cannot find a schedulable priority assignment with an arbitrary assignment A'.

VIII. Evaluation

We have conducted extensive simulation to evaluate OPMB in comparison with (i) deadline-monotonic (DM) — simple heuristic, (ii) ZSPA — the state-of-art fixed-priority assignment algorithm for general distributed real-time systems [23] — and (iii) MAA (Algorithm 2 in [11]) — the state-of-art fixed-priority assignment algorithm for a multi-domain system with a central gateway [11].

TABLE II
System model configuration

Number of buses	3 - 8
Bus type	CAN or CAN-FD
CAN bus link speed	250Kbps, 500Kbps
CAN-FD bus arbitration phase link speed	500Kbps
CAN-FD bus data phase link speed	2Mbps, 5Mbps, 8Mbps
Number of ECUs	250Kbps: 3 - 4ECUs 500Kbps: 4 - 7 ECUs 2Mbps: 7 - 10 ECUs 5Mbps: 8 - 12 ECUs 8Mbps: 10 - 15 ECUs

TABLE III
Signal characteristics

Period (ms)	share	Size (Bytes)	share
1	4%	1	35%
2	3%	2	49%
5	3%	4	13%
10	31%	5 - 8	0.8%
20	31%	9 - 16	1.3%
50	3%	17 - 32	0.5%
100	20%	33 - 64	0.4%
200	1%		
1000	4%		

TABLE IV
Configuration for signal generation

Number of signals	Number of buses × (10 - 200)
Number of destinations	1 - 4
Probability of gatewayed signal	10 – 100%

The main goals of this evaluation are to (1) compare the schedulability coverage of different algorithms for industry-size problems, (2) identify the strength of OPMB over MAA, (3) understand the reason for the identified OPMB's strength, and (4) check the feasibility of OPMB for the industry-size problems. To meet these goals, we measured schedulability coverage and execution time of each algorithm. To assess the schedulability coverage, we have designed and implemented a simulator[5] which randomly generates multi-domain system models and message sets. Generation of the system models and the message sets is detailed next.

A. Simulator Setup

1) **System model generation***:* The simulator generates a multi-domain system model based on the configuration in Table II. To scale up to industry-size problems, we use the maximum number of domains in [11] as the maximum number of buses. Our simulation adopted the CAN bus link speeds commonly used in the automotive industry. Also, the listed CAN-FD data phase speeds are supported by current commercial CAN-FD transceivers/controllers. The maximum number of ECUs on each bus is restricted to its data phase link.

[5]Available at https://github.com/TaejuPark/OPMB_RTSS_2020

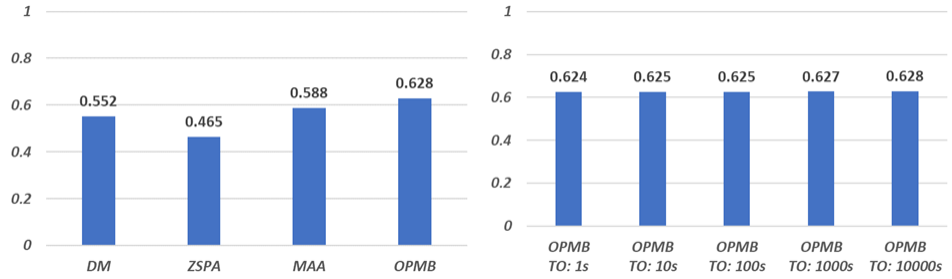

Fig. 8. (a) Schedulability coverage of the applied algorithm for 'overall'; (b) Schedulability coverage of OPMB for 'overall' with different timeouts

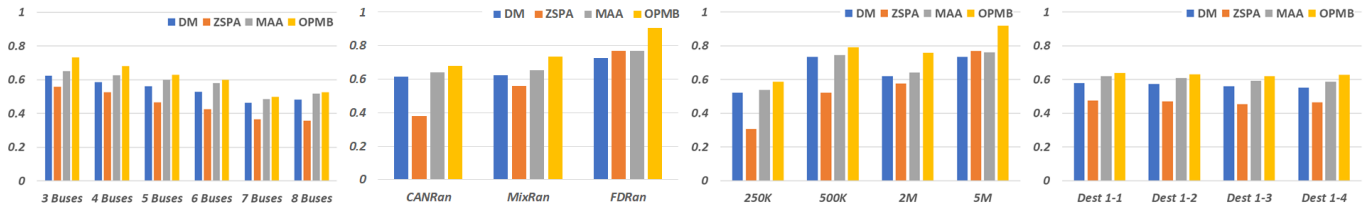

Fig. 9. Schedulability coverage of the applied algorithm for (a) fixed number of buses, (b) fixed bus-type, (c) fixed bus link speed and (4) different maximum number of destinations of a signal

2) **Message set generation***:* Our simulator generates signals (instead of messages) and packs them using Algorithm 1 in [11]. We also use the same signal characteristics used in [11] because they generate realistic in-vehicle signals based on real-world automotive benchmarks [20]. The signal characteristics are provided in Table III.

When the simulator generates a signal, the source ECU of the generated signal is randomly chosen with an equal probability. Also, the simulator determines whether the signal is forwarded to other buses or not. If a signal is determined to be forwarded to other buses, the destination ECUs are randomly chosen among all ECUs with an equal probability. Otherwise, only those ECUs that share the same bus with the source ECU of the signal are chosen as the destination ECUs for the signal.

3) **Gateway processing delay***:* We use the state-of-art analysis in [14] for CAN message processing to compute the worst-case gateway processing delay ($delay^{+}_{cgw}$):

$$T_{wait}(ISR_{rx}) + T_e(ISR_{rx}) + \sum_{i=1}^{k} \theta + T_c + T_e(Task_{tx}), \quad (6)$$

where $T_{wait}(ISR_{rx})$ is the waiting time for Rx interrupt service routine (ISR), $T_e(ISR_{rx})$ is the execution time for Rx ISR, θ is the time to compare the received ID with the ID value in the routing table, T_c is the time for converting the source bus message to the destination bus message, $T_e(Task_{tx})$ is the execution time for Tx task, and k is the number of elements in the routing table.

We set $T_e(ISR_{rx}) = 5\mu s$, $\theta = 1\mu s$, $T_c = 5\mu s$, $T_e(Task_{tx}) = 20\mu s$ according to the measurement results in [14], and set $T_{wait}(ISR_{rx}) = 0$ since the gateway is assumed to have a dedicated core for each bus in this simulation.

4) **OPMB timeout***:* Even though we try to reduce the execution time of OPMB as much as possible, its execution time could be unacceptably large. Thus, OPMB is forced to terminate upon expiration of a timer. That is, OPMB is terminated whenever the execution time exceeds a pre-defined expiration time (10,000s in this simulation) on Intel Xeon E5-2683 @ 2.10GHz, 128GB Memory. If OPMB is terminated due to a timeout, we regard it as OPMB's failure to find a schedulable priority assignment.

B. Test cases

To compare OPMB with the existing algorithms for the various system configurations, we generated test cases by randomly selecting parameters in Tables II and IV as follows:

- **Overall:** generate 36,000 cases randomly.
- **Fixed number of buses:** generate 6,000 test cases for each fixed number of buses (e.g., 6,000 cases of a 3-bus system, 6,000 cases of a 4-bus system,. . .).
- **Fixed bus-type:** generate 6,000 test cases for each fixed bus-type (CAN only, CAN-FD only, and mixed CAN/CAN-FD). Note that the system is set to have 3 bus types for these test cases.
- **Fixed bus link speed:** generate 6,000 test cases for each fixed bus link speed (250Kbps, 500Kbps, 2Mbps and 5Mbps). Note that the system is set to have 3 bus-types for these test cases.
- **Fixed the maximum number of destination ECUs:** generate 36,000 test cases for each configuration (the maximum number of destinations of a signal is 1,2,3 and 4).

Note that we only use *valid cases*, where every bus has less than 1.0 utilization (load), from the generated test cases to find a schedulable priority assignment using DM, ZSPA,

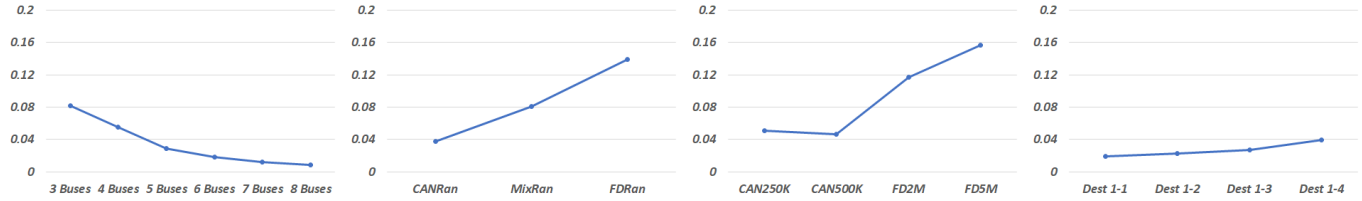

Fig. 10. Schedulability coverage gap between OPMB and MAA for (a) fixed number of buses, (b) fixed bus-type, (c) fixed bus link speed, and (4) different maximum number of destinations of a signal

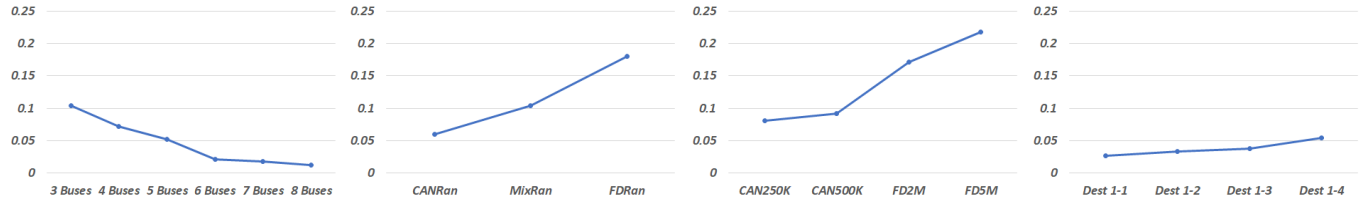

Fig. 11. Maximum room (schedulability coverage) to improve by using per-bus priority assignment over global priority assignment for (a) fixed number of buses, (b) fixed bus-type, (c) fixed bus link speed, and (4) different maximum number of destinations of a signal

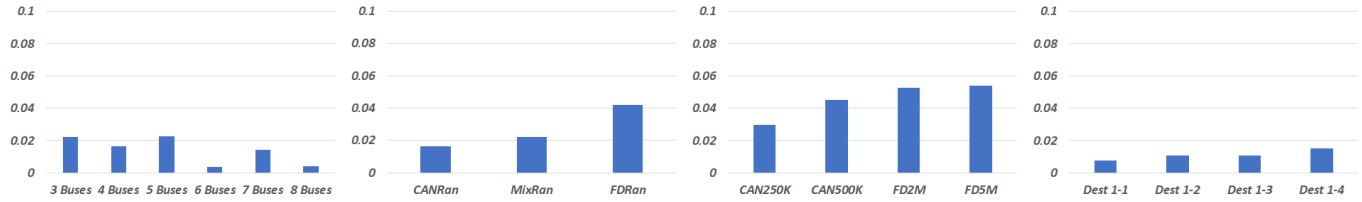

Fig. 12. OPMB timeout ratio for (a) fixed number of buses, (b) fixed bus-type, (c) fixed bus link speed, and (4) different maximum number of destinations of a signal

MAA, and OPMB. Also, when we generate the test cases, the number of buses, the average number of signals per bus and the probability of gatewayed-signals are given manually as command line arguments.[6]

C. Evaluation results and analyses

1) **Comparison of schedulability coverage:** First, we compare the schedulability coverage of the algorithms. Fig. 8(a) shows that the schedulability coverage of the evaluated algorithms for industry-size problems. As we expected, for 'overall' test cases, OPMB outperforms DM by 7%, ZSPA by 16%, and MAA by 4% in schedulability coverage. Also, Fig. 8(b) shows that even though increasing the expiration time of OPMB helps to cover more cases, the increase of schedulability coverage is not significant. Fig. 9 shows OPMB outperforming the existing algorithms regardless of system configuration.

2) **Where does OPMB have strength and why?:** The schedulability coverage gap between OPMB and MAA shown in Fig. 10. It shows that OPMB has strength for systems with a smaller number of buses, higher bus link speed, and a larger number of destinations of a signal. In fact, OPMB also has strength for systems composed of only CAN-FD buses, and the strength comes from the higher bus link speed of CAN-FD buses.

This can be reasoned about as follows: higher bus link speed and larger number of destinations of a signal make the problem more complex as it increases the available number of combinations for message priority assignment. For example, a higher link speed means a larger number of signals can be scheduled on a bus without exceeding the bus utilization limit ($= 1.0$). Also, a larger number of destinations of a signal means that a signal can have more priorities, e.g., priority 1 for bus 1, priority 2 for bus 2, . . . Thus, OPMB shows strength over MAA in these cases.

However, it is difficult to infer the reasons for strength in the case of smaller number of buses. So, we have investigated how much of room (in perspective of schedulability) to improve by using per-bus priority assignment over global priority assignment for various system configurations. If there is more room to improve for the system with a smaller number of buses, OPMB can have more chances to outperform MAA. Thus, it makes sense that OPMB has strength for systems with a smaller number of buses. To investigate the maximum room to improve, we count the following test cases since OPMB is the optimal per-bus priority assignment and MAA is the optimal global priority assignment.

- OPMB finds a schedulable priority assignment while

[6] ./run -c 3 -s 50 -p 70 means 3 buses, average 50 signals per bus, 70% of gateway signals.

TABLE V
OPMB EXECUTION TIMES (IN SECONDS)

	$t <= 1$	$1 < t <= 10$	$10 < t <= 100$	$100 < t <= 1000$	$1000 < t <= 10000$	Timeout
Schedulable	2327	3	2	5	3	0
Unschedulable	1286	14	17	7	9	56
Total	3613	17	19	12	12	56

TABLE VI
EXECUTION TIME (IN SECOND)

	DM	ZSPA	MAA	OPMB
Min	0.0004	0.0007	0.0001	0.0001
Max	0.1166	3.885	0.8056	10000
Average	0.0099	0.0912	0.0136	164.6877
Standard deviation	0.0108	0.2012	0.0343	1245.41
5% Percentile	0.0012	0.0028	0.0003	0.0005
25% Percentile	0.0032	0.011	0.0012	0.0018
50% Percentile	0.0065	0.0308	0.0035	0.0049
75% Percentile	0.0127	0.089	0.0111	0.0134
95% Percentile	0.0298	0.3641	0.0595	0.1306

MAA cannot;

- OPMB cannot find a schedulable priority assignment due to the timeout (not decidable).

The percentage of room to improve (counted cases / valid cases) is shown in Fig. 11. We can see the percentage of the maximum room to improve increases with the decrease of the number of buses and the increase of bus link speed and the number of signal destinations. This trend is exactly same as the pattern in schedulability coverage gap between OPMB over MAA. That is, the amount of benefit from OPMB is proportional to the possible room to improve by using per-bus priority assignment.

3) **Feasibility of OPMB***:* Since DM, ZSPA and MAA are polynomial-time algorithms, their execution times are expected to be small enough for industry-size (automotive) problems. In contrast, OPMB is basically an exponential-time algorithm, and hence its completion could take very long. To see the differences in execution time, we first measure the average execution time and standard deviation of each algorithm for 'overall' test cases as shown in Table VI.

The average execution times of DM, ZSPA and MAA are less than 1 second and the standard deviations are also small. Thus, about 1 second would be expected for industry-size problems. However, the average execution time of OPMB is about 164x larger than those of ZSPA and MAA, and its execution time varies widely (i.e., a large standard deviation) for the test cases. Thus, we investigate the distribution of OPMB's execution time as shown in Table V. OPMB is shown to take less than 1 second for most of the test cases (96.9% for 'overall' test cases).

We also measured the timeout ratio for the other test cases as shown in Fig. 12 with the expiration time of 10,000s. The results show that OPMB has the worst timeout ratio for systems with 5Mbps bus link speed. Because the worst timeout ratio is about 5.3%, we expect OPMB to be feasible for about 95% of real-world scenarios.

IX. EXTENSIONS

Switched Ethernet is prevalent in modern vehicles for ADAS and infotainment to handle large amounts of network traffic from camera/lidar/radar sensors. The raw data is processed and transformed into smaller-sized data and then forwarded to other (e.g., powertrain or body) domains. However, the current OPMB only covers the system composed of multiple CAN/CAN-FD buses with a central gateway, and thus cannot handle the switched Ethernet. However, OPMB can be extended to the system that includes the switched Ethernet. From OPMB's perspective, the differences between CAN/CAN-FD and switched Ethernet are (1) computing the worst-case delay in a network and (2) the limited number of priorities. We can use the timing analysis for the switched Ethernet [21] instead of Eq. (3). For the limited number (up to 8) of priorities, OPMB needs to assign the same priority to multiple messages on a network because the number of messages might be greater than 8. Thus, CLP_k should not be incremented by 1 after assigning priority to a message, but CLP_k should be incremented when there is no schedulable message with CLP_k.

X. CONCLUSION

Determining whether or not a designed in-vehicle network can meet the timing requirements of a given set of messages is important to minimize the in-vehicle network cost, thus calling for optimal priority assignment. To meet this need, we have proposed an optimal priority-assignment algorithm, OPMB, for multiple CAN/CAN-FD buses with a central gateway. It is designed based on backtracking (brute-force search with theory-proven pruning). Our in-depth simulation has demonstrated that OPMB outperforms the state-of-art priority-assignment algorithms for multi-bus systems, and has strength especially in high-speed systems which represent future automotive systems. OPMB is also shown to be feasible for most realistic automotive message sets.

ACKNOWLEDGEMENT

The work reported in this paper was supported in part by the Office of Naval Research under Grant No. N00014-18-1-2141.

REFERENCES

[1] N. C. Audsley. On priority assignment in fixed priority scheduling. *Information Processing Letters*, pages 79(1):39–44, 2001.

[2] AUTOSAR. Specification of PDU Router, AUTOSAR CP Release 4.4.0, 2018.

[3] P. Axer, M. Sebastian, and R. Ernst. Probabilistic response time bound for CAN messages with arbitrary deadlines. In *2012 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1114–1117, March 2012.

[4] E. Azketa, J. P. Uribe, M. Marcos, L. Almeida, and J. J. Gutierrez. Permutational genetic algorithm for the optimized assignment of priorities to tasks and messages in distributed real-time systems. In *2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, pages 958–965, Nov 2011.

[5] I. Broster, A. Burns, and G. Rodriguez-Navas. Comparing real-time communication under electromagnetic interference. In *Real-Time Systems, 2004. ECRTS 2004. Proceedings. 16th Euromicro Conference on*, pages 45–52, 2004.

[6] R. I. Davis, S. Kollmann, V. Pollex, and F. Slomka. Controller Area Network (CAN) Schedulability Analysis with FIFO Queues. In *2011 23rd Euromicro Conference on Real-Time Systems*, pages 45–56, July 2011.

[7] Robert I. Davis and Alan Burns. Robust priority assignment for messages on Controller Area Network (CAN). *Real-Time Systems*, 41(2):152–180, 2009.

[8] Robert I. Davis, Alan Burns, Victor Pollex, and Frank Slomka. On Priority Assignment for Controller Area Network when Some Message Identifiers Are Fixed. In *Proceedings of the 23rd International Conference on Real Time and Networks Systems*, RTNS '15, pages 279–288, New York, NY, USA, 2015. ACM.

[9] J. J. G. Garcia and M. G. Harbour. Optimized priority assignment for tasks and messages in distributed hard real-time systems. In *Proceedings of Third Workshop on Parallel and Distributed Real-Time Systems*, pages 124–132, April 1995.

[10] K. Jeffay, D. F. Stanat, and C. U. Martel. On non-preemptive scheduling of period and sporadic tasks. In *Proceedings of Twelfth Real-Time Systems Symposium*, pages 129–139, Dec 1991.

[11] Prachi Joshi, Haibo Zeng, Unmesh D. Bordoloi, Soheil Samii, S. S. Ravi, and Sandeep K. Shukla. The Multi-Domain Frame Packing Problem for CAN-FD. In *29th Euromicro Conference on Real-Time Systems (ECRTS 2017)*, 2017.

[12] U. Keskin. Evaluating Message Transmission Times in Controller Area Network (CAN) without Buffer Preemption Revisited. In *Vehicular Technology Conference (VTC Fall), 2013 IEEE 78th*, pages 1–5, Sept 2013.

[13] D. A. Khan, R. J. Bril, and N. Navet. Integrating hardware limitations in CAN schedulability analysis. In *Factory Communication Systems (WFCS), 2010 8th IEEE International Workshop on*, pages 207–210, May 2010.

[14] J. H. Kim, S. Seo, N. Hai, B. M. Cheon, Y. S. Lee, and J. W. Jeon. Gateway framework for in-vehicle networks based on can, flexray, and ethernet. *IEEE Transactions on Vehicular Technology*, 64(10):4472–4486, 2015.

[15] T. Park and K. G. Shin. Optimal priority assignment for scheduling mixed can and can-fd frames. In *2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2019.

[16] Taeju Park and Kang G. Shin. Eacan: Reliable and resource-efficient can communications. *ACM Trans. Embed. Comput. Syst.*, 18(1), February 2019.

[17] Florian Pölzlbauer, Robert I. Davis, and Iain Bate. A practical message id assignment policy for controller area network that maximizes extensibility. In *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, RTNS '16, page 45–54, New York, NY, USA, 2016. Association for Computing Machinery.

[18] R. I. Davis and A. Burns and R. J. Bril and J. J. Lukkien. Controller Area Network (CAN) Schedulability Analysis: Refuted, Revisited and Revised. *Real-Time Syst.*, 35(3):239–272, April 2007.

[19] K. W. Schmidt. Robust Priority Assignments for Extending Existing Controller Area Network Applications. *IEEE Transactions on Industrial Informatics*, 10(1):578–585, Feb 2014.

[20] Arne Hamann Simon Kramer, Dirk Ziegenbein. Real world automotive benchmark for free. In *6th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS 2015)*, 2015.

[21] D. Thiele and R. Ernst. Formal worst-case performance analysis of time-sensitive ethernet with frame preemption. In *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–9, 2016.

[22] K. W. Tindell, A. Burns, and A. J. Wellings. Allocating hard real-time tasks: An np-hard problem made easy. *Real-Time Systems*, 4(2):145–165, Jun 1992.

[23] H. Yoon and M. Ryu. Guaranteeing end-to-end deadlines for autosar-based automotive software. *International Journal of Automotive Technology*, 16(4):635–644, Aug 2015.