Optimal Priority Assignment for Scheduling Mixed CAN and CAN-FD Frames

Abstract—**Controller Area Network with Flexible Data-rate (CAN-FD) has been drawing considerable attention as the most promising substitute of Controller Area Network (CAN) thanks to its higher bandwidth, larger payload size, and physical-layer compatibility with CAN. In particular, the physical-layer compatibility allows legacy CAN-based Electronic Control Units (ECUs) to share the same communication bus with CAN-FD-based ECUs. However, CAN-based ECUs always treat a CAN-FD frame as an erroneous frame due to the difference in frame format. This, in turn, makes CAN-FD-based ECUs unable to communicate with each other via CAN-FD frames. A straightforward solution to this problem is to utilize the *silent mode* of the current CAN controller, in which a CAN node does not transmit any frame including error frames, but can receive CAN frames. However, a non-negligible time overhead is required for each mode transition, and hence degrades the schedulability of mixed CAN and CAN-FD frame sets significantly. We propose a new algorithm, called *Priority Assignment with Mode Transition* (PAMT), that minimizes the required number of mode transitions by clustering frame instances based on their frame type. Our evaluation results show that PAMT can schedule 17–18% more mixed CAN and CAN-FD frame sets than existing optimal priority assignment algorithms for CAN.**

I. Introduction

Controller Area Network (CAN) [28] is the *de facto* standard of current in-vehicle networks because of its robustness, wide deployment, low resource requirement, and real-time support. However, the advent of new functions to improve the driver's safety and comfort will make CAN unlikely to meet in-vehicle communication requirements in the near future [33]. To overcome the shortcomings of CAN, a new protocol, *Controller Area Network with Flexible Data-rate* (CAN-FD), has recently been proposed [29]. CAN-FD not only overcomes the drawbacks of CAN but also allows use of existing/legacy CAN infrastructures — e.g., Electronic Control Units (ECUs) developed with CAN controllers and transceivers, CAN wires, etc. — thanks to its physical-layer compatibility with CAN. As a result, CAN-FD has been attracting significant attention as the most promising substitute of CAN [34].

Even though CAN-FD-based ECUs can share the same communication bus with CAN-based ECUs, legacy CAN controllers, which do not support CAN-FD frame format, cause a significant problem [21]. Whenever CAN controllers receive a CAN-FD frame, they generate an error frame because the CAN-FD frame is recognized as an erroneous frame due to the difference between CAN and CAN-FD frame formats [29]. As a result, CAN-FD-based ECUs discard the CAN-FD frame upon receiving an error frame in accordance with the CAN protocol [28]. So, CAN-FD-based ECUs cannot communicate with each other via CAN-FD frames, making it impossible to realize the advantages of CAN-FD, such as relatively high bandwidth and large payload size.

Recently, hardware [3, 19] and software [21] solutions have been proposed to solve this problem. The hardware solutions [3, 19] use an additional hardware component (e.g, NXP FD Shield) which filters out CAN-FD frames before reaching the CAN controllers. However, they are more expensive and more difficult to deploy, than the software solutions. The software solution [21] is cheaper than the hardware solutions, but relies on the silent mode of CAN controller to prevent the CAN controller from generating error frames. Thus, all CAN controllers must switch their mode from normal mode to silent mode at run-time before transmitting CAN-FD frames. Also, the CAN controllers have to return to normal mode after transmitting the CAN-FD frames to resume reception of CAN frames. These mode transitions incur non-negligible time overheads and hence negatively impact the schedulability of mixed CAN and CAN-FD frame sets significantly. Thus, the existing optimal priority-assignment algorithms for CAN [5, 36] cannot find a *schedulable* priority order for the mixed frame sets even when a schedulable priority assignment exists for the mixed frame sets.

To remedy the above problem, we propose a new priority-assignment algorithm, called *Priority Assignment with Mode Transition* (PAMT), which minimizes the negative impact of the silent mode-based solution on the schedulability of a given set of mixed CAN and CAN-FD frames. PAMT reduces the required number of mode transitions for the given set of mixed frames via *type-based clustering* which groups frame instances based on their type. We prove that PAMT is an *optimal* priority assignment algorithm for mixed frame sets. We also conduct extensive simulations to evaluate the effectiveness of PAMT for mixed frame sets by comparing it with the existing optimal priority-assignment algorithms for CAN. Our simulation results show that PAMT effectively reduces the required number of mode transitions, and thus PAMT can schedule 17–18% more mixed CAN and CAN-FD frame sets than existing optimal priority assignment algorithms for CAN. This paper makes the following main contributions:

- Identify and analyze the negative impact of the software (silent mode-based) solution on the schedulability of mixed CAN and CAN-FD frame sets;
- Propose a new priority assignment algorithm, PAMT, to minimize the negative impact of using silent mode on the schedulability of mixed frame sets;
- Prove that PAMT is optimal priority assignment for mixed frame sets; and

- Demonstrate via extensive simulations that PAMT effectively reduces the required number of mode transitions and outperforms the existing optimal priority-assignment algorithms for mixed frame sets.

The rest of paper is organized as follows. Sections 2 and 3 present the background and the target system model, respectively. Sections 4 and 5 describe the problem encountered in mixed CAN and CAN-FD systems and the existing solutions, respectively. We present a new priority-assignment algorithm, PAMT in Section 6 and discuss practical issues in implementing PAMT in Section 7. Section 8 evaluates PAMT in comparison with the existing optimal priority-assignment algorithms. We discuss the related work in Section 9 and conclude the paper in Section 10.

II. CAN Primer

A. CAN Arbitration

Multiple ECUs communicate on a shared CAN bus. If two or more ECUs transmit CAN frames at the same time, the transmission order of the CAN frames is determined in a decentralized way by *bit-wise arbitration* using identifier (ID) bits in CAN frames. When an ECU transmits a CAN frame, one bit at a time, on the CAN bus, it monitors the value on the CAN bus after transmitting each bit. Especially, during the transmission of ID bits of a CAN frame, if the transmitted bit value is recessive (1) and the monitored bit value is dominant (0), the ECU loses the arbitration and stops its transmission, and will try to transmit the CAN frame again when the bus becomes idle.

B. CAN Controller Modes

Even though the CAN protocol [28] does not specify the operation modes of CAN controller, the commercial CAN controllers have similar operation modes. Below we briefly introduce the two common operation modes.

Normal Mode is the basic operation mode. In this mode, there is no limitation for an ECU to play a role as transmitter or receiver. Thus, an ECU can transmit any type of frame (i.e., any of data, remote, error, and overload frame) and can also generate an acknowledgement (ACK) signal. In this mode, the error (transmit, receive) counters in a CAN controller are activated.

Silent Mode is often called *listen-only mode*. In this mode, an ECU cannot transmit any CAN frame including an error frame and ACK signal on the CAN bus. However, the ECU can receive any message on the CAN bus. Thus, this mode is usually used by an application which requires bus monitoring. In this mode, the error counters are deactivated.

III. Mixed CAN and CAN-FD System Model

A. Network Model

We consider a system consisting of multiple ECUs connected via one shared CAN bus as illustrated in Fig. 1. Some of the ECUs are equipped with CAN-FD controllers and transceivers, while the others are equipped with CAN controllers and transceivers. For convenience, we will use the term 'CAN-FD node' ('CAN node') to represent an ECU equipped with a CAN-FD (CAN) controller and a CAN-FD (CAN) transceiver.

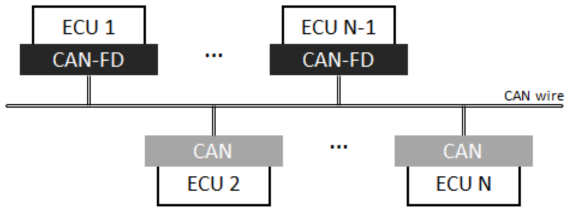

Fig. 1. Network Model

We say the system is in 'CAN mode' when the CAN controllers are in normal mode because only CAN frames are transmitted on the bus in this mode. Likewise, we say the system is in 'FD mode' when the CAN controllers are in silent mode. To avoid transmitting CAN-FD frames in CAN mode, CAN-FD node does not enqueue the CAN-FD frames in CAN mode. Because our approach send a special message to trigger mode transition of CAN controllers, CAN-FD nodes easily know the system state. **S**ystem mode

Some may expect this mixed CAN and CAN-FD network architecture to be short-lived and used only during the transition from CAN to CAN-FD, but this architecture would last for a long time, because (1) a vehicle platform, once developed, is utilized for a long time[1]; (2) the implementation cost of a CAN node is cheaper than that of a CAN-FD node and the automotive industry seldom uses anything more than absolutely needed to save cost; and (3) automotive manufacturers tend to use already-verified modules/systems to meet the mandatory requirements, such as safety.

B. Mixed Frame Model

We adopt the widely-used CAN frame model in [9, 18] with an additional parameter indicating whether a frame is CAN or CAN-FD frame. Thus, a frame is defined as $F_i = \{T_i, D_i, J_i, FD_i, C_i\}$ where T_i is the period of F_i, D_i the relative deadline of F_i; J_i the release jitter of F_i; FD_i the type of F_i, $FD_i \in \{0, 1\}$ – if $FD_i = 0$ then F_i is a CAN frame else F_i is a CAN-FD frame; C_i is the transmission time of F_i and depends on the data length of F_i and FD_i.

C. Mixed Frame-Instance Model

F_i^j is an instance of frame F_i, and hence inherits the properties of F_i such as frame type and transmission time. Thus, F_i^j is defined as $F_i^j = \{A_i^j, D_i^j, J_i, FD_i, C_i\}$ where A_i^j is the release time of F_i^j and D_i^j is its deadline such that $D_i^j = A_i^j + D_i$.

We will assign priorities to frame instances in an mixed frame instance set I:

$$I = \{F_i^j | \forall i, j \ A_i^j < HP\}$$

[1]According to a report by the Center for Automotive Research, a developed platform lasts over 5 years on average (http://www.cargroup.org/automotive-product-development-cycles-and-the-need-for-balance-with-the-regulatory-environment/)

where $HP = LCM\{T_1, \ldots, T_n\}$ is the planning cycle (the least common multiple of periods) of a given mixed frame set.

We use f_i to represent a frame instance of priority i; f_i has a higher priority than f_j if $i < j$, i.e., the lower the number, the higher the priority.

IV. Scheduling Mixed CAN and CAN-FD

A. Why Problem?

According to the CAN-FD specification [29], the format of CAN-FD data frame is partially different from that of CAN data frame to support higher bandwidth and larger payload size as shown in Fig. 2. Due to this frame format difference, CAN nodes always recognize CAN-FD frames as erroneous frames (CRC error) and generates an error frame whenever they receive a CAN-FD frame. Because of this incorrect error detection, an 'innocent' CAN-FD frame will be retransmitted by the sender and the retransmitted CAN-FD frame will again be detected as an erroneous frame as illustrated in Fig. 3 (Left). This makes the communication between CAN-FD nodes via CAN-FD frames impossible, thus losing all the advantages of CAN-FD. Both hardware [3, 19] and software (silent mode-based) [21] solutions to this problem have been proposed recently.

B. Hardware Solution

The hardware solutions [3, 19] require an additional hardware component which filters out the CAN-FD frames by inspecting FDF (FD Format) bit of all incoming frames in front of the CAN controller as shown in Fig. 3 (Right). Thus, CAN-FD nodes can communicate with each other using CAN-FD frames without any software change. However, the specialized hardware has to be attached to all the CAN nodes, increasing implementation and deployment costs. A single hardware component might be inexpensive, but the cost for its mass production could be significant. Note that more than 100 million new vehicles are built and sold each year [1].

C. Software Solution

The software solution [21] relies on the silent mode of the legacy CAN controllers. Before transmitting a CAN-FD frame, an ECU has to transmit a special CAN frame (trigger frame) that triggers a mode transition. All CAN nodes must transit from normal mode to silent mode upon receiving a trigger frame. A CAN-FD node then begins the transmission of CAN-FD frames as shown in Fig. 3 (Mid). Since all the CAN nodes are in silent mode, CAN-FD nodes can communicate with each other using CAN-FD frames. A CAN-FD node must thereafter send another trigger frame to wake up the CAN nodes from silent mode to normal mode as shown in Fig. 3 (Mid). Although the software solution can resolve the problem without any additional hardware, it incurs non-negligible delays for mode transitions.

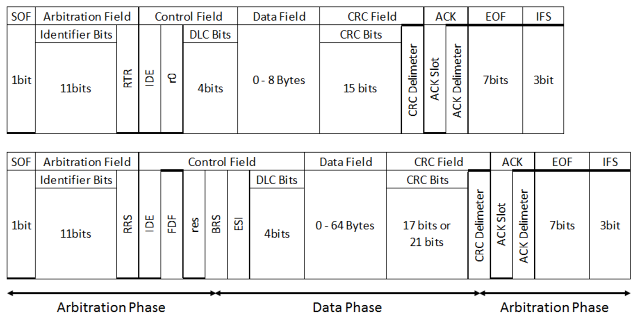

Fig. 2. Format of CAN data frame (Top) and format of CAN-FD data frame (Bottom)

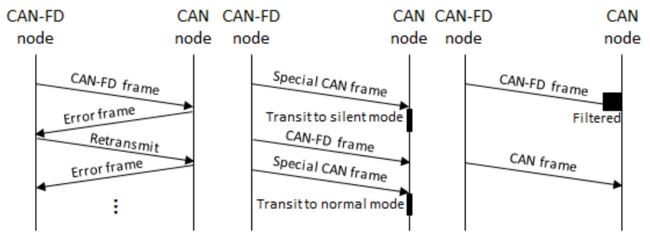

Fig. 3. (Left) Problem in scheduling mixed CAN and CAN-FD frames; software (Mid) and hardware (Right) solutions

Analysis of Mode-Transition Delay: The time overhead of the software solution for a mode transition consists of two parts; *transmission time* of a trigger frame and *processing time* of a mode transition.

The transmission time of a trigger frame depends on the CAN bus speed as well as its payload size. We must thus define and use the format of a trigger frame to compute its transmission time. So, a trigger frame is differentiated from a normal data frame by assigning it a unique ID (of 11 bits) and its payload size is set to 0. With this setting, if the CAN bus speed is 500Kbps then the transmission time of the trigger frame is $112\mu s$.

The processing time of a mode transition depends on the computing power of a CAN node. However, since a mode transition is very simple (writing a value to a register in the CAN controller and reading the changed value in the register), there will be only a small processing time variation. We have measured the processing time on our experimental platform as shown in Fig. 4. We use Arduino [4] and MCP2515 CAN controller [20] to build a CAN node. The processing time was about $84\mu s$ on average, and hence a mode transition takes about $200\mu s$ in total (500Kbps CAN bus speed).

Negative Impact of Time Overhead: Fig. 5 illustrates the negative impact of the time overhead of a mode transition;

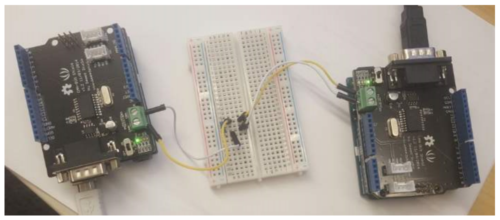

Fig. 4. Experimental platform

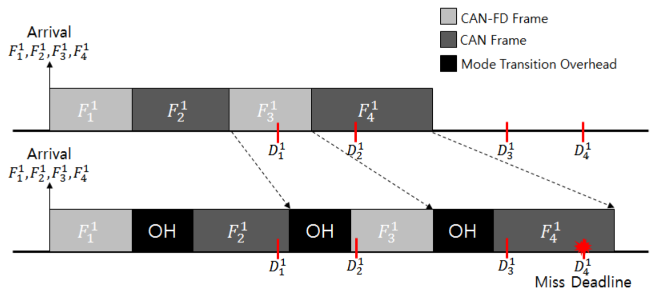

Fig. 5. Due to the time overhead, the delivery/completion time of a given frame increases and a frame misses its deadline

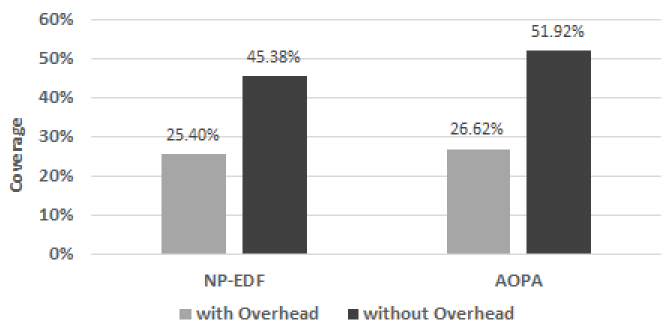

Fig. 6. The coverage of existing optimal priority assignment with or without the mode-transition overhead for given mixed frame sets.

the delivery/completion times of F_2^1, F_3^1, and F_4^1 increase, missing the deadline of F_4^1. That is, the time overhead degrades the schedulability of a given mixed frame set. According to our simulation results (in Section 7), more than 20% of given mixed frame sets become unschedulable due to the overhead with the existing frame-level optimal priority assignment (AOPA [5]) and frame-instance-level optimal priority assignment (NP-EDF [15]) as shown in Fig. 6.

V. Problem Statement

Placing CAN and CAN-FD nodes on the same network to utilize the established CAN infrastructure is problematic as we discussed earlier, and thus hardware and software solutions have been proposed to solve the problem. The software solution is more attractive than the hardware solution for a cost reason, but it degrades the schedulability of mixed CAN and CAN-FD frame sets due to its reliance on the mode transitions of the existing CAN controllers. As a result, the schedulability or coverage of the existing optimal priority assignment degrades significantly, i.e., the software solution with the existing optimal priority assignment fails to schedule many mixed frame sets while meeting all of their frame deadlines.

In order to enable the software solution to schedule more mixed frame sets, we propose a new priority-assignment algorithm, called *Priority Assignment with Mode Transition* (PAMT), that minimizes the schedulability degradation by minimizing the mode-transition overhead.

Frame	T_i	D_i	J_i	FD_i	C_i
F_1	5ms	0.75ms	0ms	0	272μs
F_2	5ms	1ms	0ms	1	320μs
F_3	5ms	1.5ms	0ms	0	272μs
F_4	5ms	1.75ms	0ms	1	400μs

TABLE I
An example frame set

VI. Priority Assignment with Mode Transitions

We now present PAMT for a given mixed frame instance set, which minimizes the coverage loss of the software solution. We first introduce the basic idea of PAMT and then provide its details.

A. Basic Idea of PAMT

Non-Preemptive Earliest Deadline First (NP-EDF) based priority assignment and *type-based clustering* are the key of PAMT.

PAMT assigns priorities to the frame instances in a given mixed frame set based on NP-EDF, because NP-EDF is known to be optimal for work-conserving system like CAN [15] if there were no mode transition overhead. However, the software solution may incur a high mode-transition overhead, and hence PAMT performs type-based clustering of frame instances to reduce the mode-transition overhead. For example, suppose that priorities are assigned to the frame instances of a given mixed frame in Table I. PAMT first assigns priorities to the frame instances based on NP-EDF, but this incurs 3 mode transitions, causing F_4^1 to miss its deadline as shown in Fig. 7 (Top). To reduce mode transitions and to make the given mixed frame set schedulable, PAMT performs type-based clustering. As illustrated in Fig. 7 (Bottom), after the type-based clustering, the same-type frame instances are clustered and the number of mode transitions is reduced to 1. As a result, there are no deadline misses with the clustered priority ordering, making the given mixed frame set schedulable.

B. PAMT Algorithm

Even though the type-based clustering is a natural way to reduce mode-transition overheads, it is challenging to group frame instances so as to minimize the degradation of schedulability, because the type-based clustering can increase

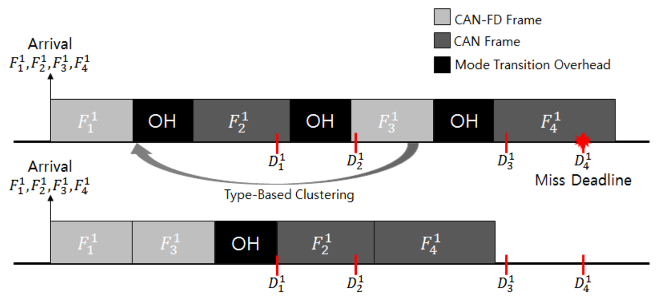

Fig. 7. (Top) Assign priorities to frame instances based on NP-EDF; (Bottom) reducing mode-transition overheads via type-based clustering

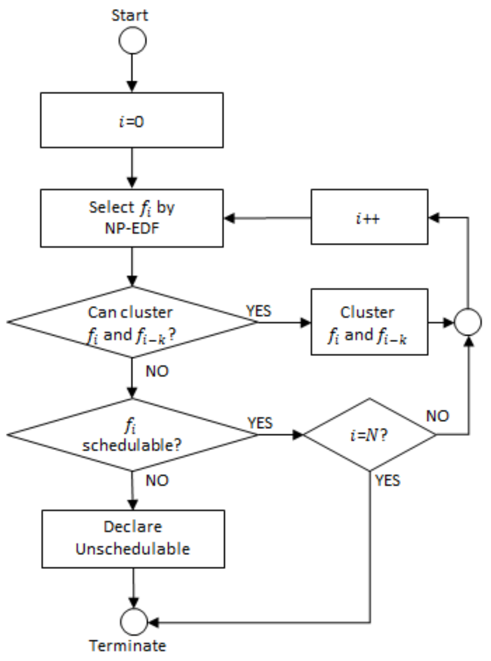

Fig. 8. Flowchart of PAMT

the delivery/completion times of frame instances (e.g., F_2^1 in Fig. 7 (Bottom)), which can cause unexpected deadline misses.

Next, we will first give an overview of PAMT algorithm that meets the above challenge, and then give a detailed account of each part of the algorithm.

Algorithm Overview: Fig. 8 shows how PAMT operates on a given mixed frame instance set I. PAMT selects a frame instance according to NP-EDF and assigns priority i to the frame instance as the first step. After selecting the frame instance (f_i), PAMT checks several conditions to cluster f_i and f_{i-k}, where f_{i-k} is the nearest frame instance whose type is the same as the type of f_i. If all conditions are met, PAMT performs the type-based clustering. Otherwise, PAMT does not perform the type-based clustering but just checks whether f_i meets its deadline or not. If f_i meets its deadline, PAMT selects another frame instance to assign priority $i+1$ according to NP-EDF. Otherwise, PAMT declares the given mixed frame instance set I unschedulable and terminates the process. This process will be repeated until all the frame instances in I are assigned priorities. The pseudo-codes of PAMT implementation are stated in Algorithm 1 and 2 in Appendix section. Next, we will detail how to implement each procedure.

Select f_i according to NP-EDF: We need an off-line assignment of priorities to the frame instances in a mixed frame set I even though NP-EDF is an on-line scheduling algorithm. So, we simulate a planning cycle or hyper-period (HP) of I to assign priorities to the frame instances in I. Usually, the periods of in-vehicle CAN frames are harmonic [24], and thus an HP is not too long to simulate. According to our measurements through on-board diagnostic (OBD) port,

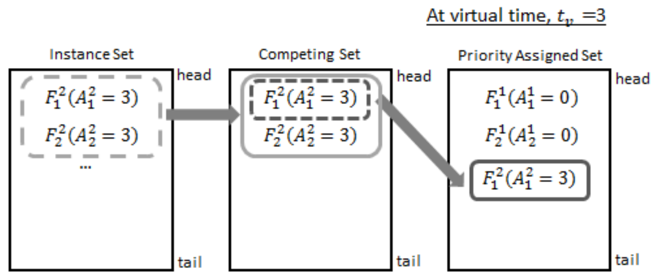

Fig. 9. Assign priority to a frame instance based on NP-EDF

HP is 6s for 2015 Chevrolet Trax LT AWD.

To simulate an HP, we manage a virtual time t_v and 3 data structures: *mixed-frame instance set* (I), *competing set*, *priority-assigned set*. This simulation requires I to be sorted in ascending order of frame instances' arrival times and the virtual time to be initialized with 0. Described below is how the HP is simulated.

At time t_v, if the arrival times of frame instances in I are earlier than, or equal to t_v, then the frame instances are migrated from I to the competing set as shown in Fig. 9. After the migration, PAMT sorts the competing set in ascending order by the deadline, and then selects the frame instance with the earliest deadline from the competing set to assign priority i. If two or more frame instances have the same earliest deadline, PAMT selects one of them that has the same type as f_{i-1} to avoid a mode transition between f_{i-1} and f_i. The chosen frame instance is then moved to the tail of the priority-assigned set as shown in Fig. 9. Note that the index of a priority-assigned set indicates the priority of a frame instance.

Cluster f_i and f_{i-k}: Suppose PAMT selects the frame instance (F_p^q) to assign priority i and f_{i-1} has a different type from F_p^q. Let f_{i-k} be the nearest frame instance which has the same type as F_p^q (f_i), and both f_{i-k} and F_p^q belong to the same busy period as shown in Fig. 10. Clearly, promoting the priority of F_p^q to $i-k+1$ reduces the number of mode transitions since it eliminates the mode transition between F_p^q and f_{i-1}. However, this priority promotion is not always possible. All of the following conditions must be met for the priority promotion:

C1: $f_{i-1}.type \neq f_i.type$

C2: After promoting the priority of f_i to $i-k+1$, $d_p \geq e_p, \forall p \in \{i-k+1,\ldots,i\}$

C3: $a_i \leq a_{i-k+1}$ or $a_i \leq e_{i-k}$

where a_i is the arrival time of f_i, d_i is the deadline of f_i, and e_i is the completion time of f_i.

C1 is obvious, and hence its discussion is omitted. If we promote the priority of f_i to $i-k+1$, then it will delay the completion of frame instances between f_i and f_{i-k}. If any of these delayed frame instances violates its deadline, then the priority promotion is not allowed. That is, the delayed frame instances must finish before their deadlines and **C2** must hold.

The last condition **C3** comes from the unique characteristic of CAN scheduling, non-preemptive work-conserving schedul-

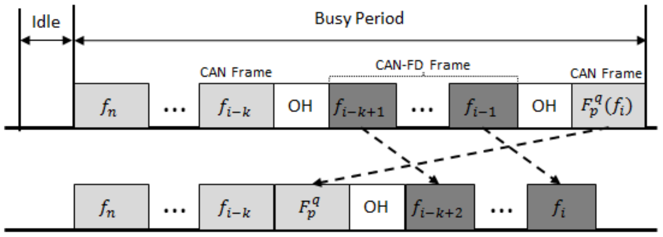

Fig. 10. Type-based clustering. Promoting the priority of f_i to $i-k+1$ to reduce the mode-transition overhead

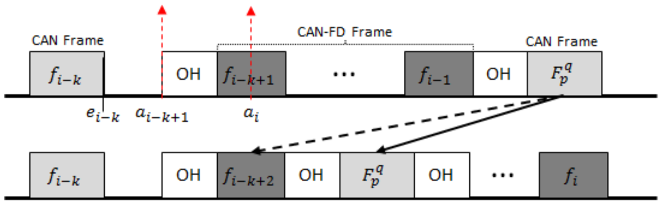

Fig. 11. Violation of **C3**. ($a_i > a_{i-k+1}$ and $a_i > e_{i-k}$)

ing. Suppose $a_i > a_{i-k+1}$ and $a_i > e_{i-k}$ as shown in Fig. 11. We expect F_p^q to be scheduled right after the transmission of f_{i-k} by promoting the priority of F_p^q to $i-k+1$ (dotted line). However, f_{i-k+2} is scheduled before transmission of $F_p^q(f_{i-k+1})$ (solid line), since there is no ready frame instance at e_{i-k} and f_{i-k+2} arrives before F_p^q arrives. So, clustering f_i and f_{i-k} is impossible even though the priority of F_p^q is promoted to $i-k+1$. The priority promotion in this case could rather increase the number of mode-transitions as shown in Fig. 11.

After executing the above procedures, the virtual time is updated to the completion time of the lowest-priority frame instance. If there is no frame instance to be moved from the mixed frame instance set to the competing set and if the competing set is empty, then the arrival time of the frame instance at the head of the instance set is assigned as the next virtual time. For example, if the arrival time of the frame instance at the head of the instance set is A_p^q, then the virtual time becomes A_p^q.

C. Optimality of PAMT

We now prove that PAMT is the optimal priority assignment for a given mixed frame instance set. That is, if PAMT cannot schedule a given mixed frame instance set, then no other priority assignment algorithm can find a schedulable priority order for the mixed frame instance set.

Lemma 1. *Let I_A be the set of frame instances in a busy period A. PAMT minimizes the number of mode transitions for I_A if no deadline miss is allowed.*

Proof: Let S_k ($S_k \subset I_A$) be the set of chosen frame instances whose cardinality is k. We will show that PAMT minimizes the number of mode transitions for S_k regardless of k if no deadline miss is allowed. This way, we can prove Lemma 1 because S_k is the same as I_A if $k = |I_A|$.

1. (Initial, $k = 1$). Since there is only one frame instance, there is no mode transition.

2. (Suppose this holds for $k = i-1$). Assume that PAMT minimizes the number of mode transitions for S_{i-1} without any deadline miss.

3. (Show this holds for $k = i$). Let g be the i^{th} frame instance chosen by PAMT. To avoid any additional mode transition, we need to schedule g right after the transmission of a frame instance whose type is the same as that of g. Since PAMT selects frame instances according to NP-EDF, we only consider placing g right after f_{i-k} which is the latest same-type frame instance in S_{i-1}. Let's analyze the following three cases.

Case 1. Suppose the arrival time of g is earlier than, or equal to e_{i-k} and placing g right after f_{i-k} does not cause any deadline miss. Then, we need to show that PAMT moves g to right after f_{i-k}. We know that **C**3 ($a_i \leq e_{i-k}$) and **C**2 holds by supposition. If **C**1 is not met, then $f_{i-k} = f_{i-1}$. So, g is already placed at right after f_{i-k}. Otherwise, all three conditions are met and PAMT moves g to right after f_{i-k} by performing type-based clustering. Thus, g doesn't incur any additional mode transition, and hence PAMT minimizes the number of mode transitions.

Case 2. Suppose the arrival time of g is larger than e_{i-k}. For this case, we need to show that additional mode transitions incurred by g are unavoidable and only one additional mode transition is incurred by g under PAMT. Since all the frame instances scheduled after f_{i-k} have different types from g, the type of the frame instance scheduled right before g is different from that of g. Thus, the additional mode transition before transmitting g is unavoidable. In this case, PAMT does not perform type-based clustering due to the violation of **C**3. Instead, PAMT schedules g last, i.e., there is no frame instance after g and the number of mode transitions incurred by g is 1. So, PAMT minimizes the number of mode transitions.

Case 3. Suppose the arrival time of g is earlier than, or equal to e_{i-k} and placing g right after f_{i-k} causes at least one deadline miss. In this case, we need to show that the additional mode transitions incurred by g are unavoidable and only one additional mode transition is incurred by g under PAMT. Since placing g right after f_{i-k} causes at least one deadline miss, g must be scheduled after f_{i-k+1} to avoid any deadline miss. However, the frames instances (from f_{i-k+1} to f_{i-1}) have different types from g. Thus, the additional mode transition before transmitting g is unavoidable. In this case, PAMT does not perform type-based clustering due to the violation of **C**2. Instead, PAMT schedules g last, i.e., there is no frame instance after g and the number of mode transitions incurred by g is 1. Thus, PAMT minimizes the number of mode transitions. □

Theorem 1. *PAMT is the optimal priority-assignment algorithm for a mixed frame instance set I.*

Proof: We prove this theorem by induction. Let K be the number of priority-assigned frame instances and let g_i be the i^{th} frame instance chosen by PAMT.

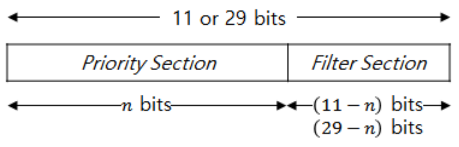

Fig. 12. Separation of CAN ID into priority and filter sections

1. (Initial, $K = 1$). Since there is only one frame instance, every priority-assignment algorithm is optimal.

2. (Suppose this holds for $K = i - 1$). Assume that PAMT is the optimal for $I_{i-1} = \{g_1, \ldots, g_{i-1}\}$.

3. (Show this holds for $k = i$). We will show that PAMT is optimal for the mixed frame instance set $I_i = \{g_1, \ldots, g_i\}$ by proving that there is no schedulable priority order for I_i if PAMT declares I_i unschedulable. Let's consider the following two cases.

Case 1. The type of g_i is the same as that of f_{i-1}. In this case, PAMT schedules g_i last (after f_{i-1}) because **C1** is not met. If g_i meets its deadline, PAMT makes I_i schedulable. However, if g_i misses its deadline, we need to show that there is no schedulable priority order for I_i. Suppose g_i misses its deadline. There are two ways to make g_i schedulable: (1) reduce the number of mode transitions during a busy period in which g_i resides; (2) schedule g_i earlier than last. By Lemma 1, PAMT minimizes the number of mode transitions in a busy period, so there is no way to reduce the number of mode transitions. Thus, we only need to consider (2). Since PAMT selects a frame instance according to NP-EDF, the frame instances, which are transmitted after the arrival of g_i, have earlier deadlines than g_i. This means that scheduling g_i at any possible instant causes at least one deadline miss. Thus, there is no schedulable priority order for I_i, and hence PAMT is optimal.

Case 2. The type of g_i is different from that of f_{i-1}, satisfying **C1**. Thus, if both **C2** and **C3** are met, PAMT schedules g_i right after f_{i-k} which is the nearest same-type frame instance in I_{i-1} and every frame instance in I_i meets its deadline (**C2**), making PAMT optimal. If either **C2** or **C3** is not met, PAMT schedules g_i last. If g_i meets its deadline last, every frame instance meets its deadline, thus making PAMT optimal. If g_i misses its deadline, we need to show that there is no schedulable priority order for I_i. As in **Case 1**, there are two ways to make g_i schedulable and we only need to consider the second case by Lemma 1. Also, like **Case 1**, scheduling g_i at any possible instant causes at least one deadline miss because the frame instances transmitted after the arrival of g_i have earlier deadline than g_i. Thus, there is no schedulable priority order for I_i, hence making PAMT optimal. □

VII. Practical Issues

A. Assigning ID to frame instances

A CAN controller does not, in practice, accept all incoming CAN frames to reduce the processing load of the host ECU [25]. The CAN controller filters the incoming CAN frames by comparing the IDs of incoming frames with the registered

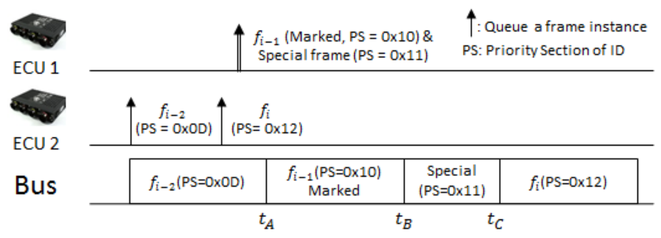

Fig. 13. Insert a special CAN frame

values in its receive filter. However, the number of receive filters is limited in a commercial CAN controller (e.g., only 6 filters in MCP2515 [20]). The limited number of receive filters makes difficult to implement PAMT because PAMT assigns priorities to frames instances, not frames. For example, when a frame F_1 is instantiated three times in a planning cycle, these three different instances have different priorities/IDs. Then, to receive all the instances F_1, an ECU has to register all of the three IDs in its receive filters.

To resolve this problem, we separate a CAN ID into *priority* and *filter* sections, as shown in Fig. 12. The priority section is only used to distinguish the priority of CAN frames. Thus, the bits in the priority section are set as '*don't care bits*' in the mask registers[2]. Since the priority section is forwarded to the filter section, it can be used in the ID arbitration process. That is, the lower the number in the priority section, the higher the priority in CAN frame scheduling. We can set the priority determined by PAMT in the priority section directly. The filter section is only used for filtering the incoming CAN frames. We give a unique value to each frame (not frame instance) and put the value in the filter section. For example, we give a value of 1 to F_1 and put the value of 1 to the filter section of the instances of F_1. As a result, an ECU can receive all the instances of F_1 by using only one (not multiple) receive filter(s).

Let n be the number of bits in the priority section with which all possible priorities must be covered. PAMT assigns a unique priority to each frame instance in a given mixed-frame set I. We also reserve whole odd numbers for a special CAN frame to trigger a mode transition. A frame instance can only have an even-numbered priority. For example, the priority of f_{i-1} is 0x10 and that of f_i is 0x12. 0x11 is reserved for the trigger frame (see Section 7.2). Hence, 2^{n-1} has to be larger than the number of frame instances in I and we select the minimum n that satisfies this requirement. Also, 2^{11-n} or 2^{29-n} has to be larger than the number of frames in a given mixed-frame set because we need to assign a unique value to each frame.

B. Triggering a Mode Transition

Triggering a mode transition precisely at the specified time is important because a late/early mode transition can cause severe problems. For example, a CAN node will generate an error frame when the node receives a CAN-FD frame due

[2]When the CAN controller performs bitwise comparisons, it ignores several bits which are set to the *'don't-care bits'* in a mask register.

to a late transition (from normal to silent). Also, a CAN frame instance, which is sent by a CAN-FD node, may not be delivered to a CAN node due to the late transition (from silent to normal). To transmit a trigger frame at a precise time, we mark frame instances after which a trigger frame must be transmitted. For example, if f_{i-1} is a CAN frame and f_i is a CAN-FD frame, then we mark f_{i-1}. Since frame instances in the priority-assigned set are sorted in their transmission order, we can easily determine which frame instances should be marked.

At runtime, if an ECU queues a marked frame instance in its transmission buffer (TxObject), it also queues a trigger frame in TxObject. To transmit the trigger frame right after the marked frame instance, the value in the priority section of the trigger frame is larger by 1 than that of the marked frame instance. For example, as shown in Fig. 13, the value in the priority section of a marked frame instance (f_{i-1}) is 0x10 and that of the corresponding trigger frame is 0x11. This way, the ID of a trigger frame can be larger than that of any other frame instances queued in TxObject. Thus, the trigger frame can be transmitted right after the transmission of the corresponding marked frame instance by winning the CAN bus arbitration.

C. Transient Error

Rare transient errors (bit error) can occur on CAN bus due to electromagnetic interference (EMI). Since each transient error is handled by generating error frames and retransmitting the unsuccessful CAN frame, the error causes additional delays to the delivery/response time of CAN frames. That is, the frame instances scheduled under PAMT may miss their deadlines due to the transient errors. To account the transient errors, we compute the maximum possible delay to f_i caused by the transient error (α_i^{tc}) and reflect the delay into the deadline.

$$\alpha_i^{tc} = \eta(\lambda, R, d_i) \times E$$

where λ is the maximum transient error rate determined during the design phase of a vehicle according to the knowledge of the worst environment in which the vehicle operates [23], R is the reliability requirement of a vehicle system, $\eta(\lambda, R, d_i)$ is the maximum number of transient errors possible within d_i, and E is the error recovery time [8]. Here, we compute $N = \eta(\lambda, R, d_i)$ under the assumption that the process of transient errors is Poisson, as commonly used for CAN transient errors [8, 6]:

$$\begin{aligned} N = \quad & \underset{Z_m}{\text{argmin}} \quad 1 - \sum_{Z=0}^{Z_m} p(Z, d_i) \\ & \text{subject to} \quad 1 - \sum_{Z=0}^{Z_m} p(Z, d_i) \le R. \end{aligned} \tag{1}$$

where $p(Z, d_i)$ is the probability of Z errors within d_i.

$$p(Z, t) = \frac{e^{-\lambda t}(\lambda t)^Z}{Z!}. \tag{2}$$

D. Unsynchronized Clock

Since there is no global clock on CAN, CAN frames are triggered according to the local time clock of each ECU. However, the local clocks are not synchronized with each other, and thus there is a clock drift/skew between ECUs. Unfortunately, this clock drift may alter the scheduling order of messages at runtime. For example, if the maximum drift on an ECU is δ and f_i and f_j ($i < j$) are sent by different ECUs and $a_i + \delta > a_j - \delta$, f_j could be transmitted before f_i.

Because the runtime change in scheduling order can incur an additional delay to the delivery/response time of CAN frames, the frame instances scheduled under PAMT may miss their deadlines. Thus, as in the previous subsection, we compute the maximum possible delay to f_i caused by the unsynchronized clocks (α_i^{uc}) and account for the delay in the deadline:

Here we assume that a synchronization protocol [30] for CAN is applied, and thus the maximum drift (δ) is limited and the arrival time range of f_i is also limited; $a_i \in [a_i - \delta, a_i + \delta]$. According to [30], the maximum drift can be limited by 20 μs, which is much smaller than the transmission time of a CAN frame. So, we assume that $\delta < \min_{i \in F} C_i$ and $\delta < E$.

1) Finding ζ_i: Suppose $I_{i,rev} = \{f_j | i < j \ \& \ a_i + \delta \ge a_j - \delta\}$. Then, the frame instances in $I_{i,rev}$ can be transmitted before transmitting f_i due to the clock drift at runtime unlike the deterministic scheduling order by PAMT.

$I_{i,rev,diff}$ is the set of frame instances whose type is different from the type of f_i in $I_{i,rev}$, and $I_{i,rev,same}$ is the set of frame instances whose type is the same as that of f_i in $I_{i,rev}$.

Lemma 2. *f_i is always transmitted before f_j if $f_j \in I_{i,rev,diff}$.*

proof: Suppose the type of f_i is different from that of f_j. Then, there is at least one marked frame instance f_k ($i \le k < j$) to trigger a mode transition. If $a_k + \delta > a_j - \delta$, f_j can be queued before queuing f_k. However, the mode is not changed yet at the arrival of f_j. Thus, f_j cannot be transmitted before f_k because a FD frame cannot be queued in the CAN mode and a CAN frame cannot be transmitted in the FD mode. If f_i arrives earlier than f_k, f_i is transmitted before f_j. If f_k arrives earlier than f_i (in the case of $f_i + \delta > f_k - \delta$), f_k can be transmitted before f_i. Since $\delta < \min_{i \in F} C_i$, f_i is guaranteed to arrive during the transmission of f_k. Thus, f_i is transmitted right after the transmission of f_k. Hence, f_i is transmitted before f_j.

Lemma 3. *The maximum delay to f_i caused by the reversed order between f_i and $f_j \in I_{i,rev,same}$ is $\max_{f_j \in I_{i,rev,same}} c_j + \delta$ where c_j is the transmission time of f_j.*

proof: Suppose the type of f_i is the same as that of f_j, and f_j is transmitted before f_i at runtime due to the unsynchronized clocks. Because f_i is guaranteed to arrive during the transmission of f_j, f_i is always transmitted right after the transmission of f_j. The worst-case scenario which contributes the maximum delay to f_i is that f_j arrives at $a_i + \delta - \epsilon$ and f_i arrives at $a_i + \delta$ where ϵ is a very small. In the worst case, the delay to f_i is

$c_j + \delta$. Thus, the maximum delay to f_i caused by the reversed order between f_i and $f_j \in I_{i,rev,same}$ is $\max_{f_j \in I_{i,rev,same}} c_j + \delta$.

Collorary 1. *α_i^{uc} is $\max_{f_j \in I_{i,rev,same}} c_j + \delta$*

proof: By Lemma 2, if $f_j \in I_{i,rev,diff}$, f_i is always transmitted before f_j. In such a case, the maximum delay of f_i due to the unsynchronized clocks is δ. Since $\delta \leq \max_{f_j \in I_{i,rev,same}} c_j + \delta$ (by Lemma 3), α_i^{uc} is $\max_{f_j \in I_{i,rev,same}} c_j + \delta$.

E. Sporadic Frames

As described in Section III.C, frames arrive periodically. However, in practice, some frames can be triggered by asynchronous events or vehicle conditions, and arrivals of such messages can be represented with a sporadic frame model (with minimum inter-arrival times). Since our approach is designed with a periodic model, sporadic frames should be converted to periodic frames by using their minimum inter-arrival time as the period. Suppose, for example, transmitting a message which includes brake pedal pressure is triggered by an event that a vehicle driver is pressing the brake pedal. This message should be converted to a periodic message. Thus, a null message (when the driver does not push the brake pedal) or a message that contains the brake pedal pressure (when the driver presses the brake pedal) should be transmitted periodically.

VIII. Evaluation

We have conducted extensive simulations to evaluate PAMT in comparison with NP-EDF, optimal frame-instance level priority assignment, and Audsley's Optimal Priority Assignment (AOPA), the well-known optimal frame-level priority assignment. We focus on the schedulability degradation of each priority assignment algorithm by measuring its coverage. We also measure the coverage of the optimal priority assignments when we use the hardware-based solution by ignoring the mode-transition overhead (labeled with AOPA* and NP-EDF*). The coverage of the hardware-based solution is the best achievable because AOPA and NP-EDF are proven to be optimal for CAN scheduling. We also evaluate PAMT-R, which accounts for transient errors (Section VII.C) and unsynchronized clocks (Section VII.D) by using a virtual deadline $d_{i,r} = d_i - \alpha_i^{tc} - \alpha_i^{uc}$ instead of d_i. To compute $d_{i,r}$, we set $\lambda = 0.01/s$, $R = 2.6 * 10^-9/s$ (SIL in IEC-61508 [2]) and $\delta = 20\mu s$.

A. Simulation Setup

The Benchmark for Simulations: We use NETCARBENCH [7] (powertrain configuration), which is a widely-used CAN benchmark. Since its latest version does not yet support the CAN-FD frame, we slightly modified NETCARBENCH to support CAN-FD. If the payload of a generated frame is larger than 8, then the type of the frame is CAN-FD. Otherwise, we assign the frame type randomly. For our simulation, we generated 10,000 mixed frame sets from NETCARBENCH. We assume that the jitter of each frame is 0 and the transmission of all the frames begins at time 0.

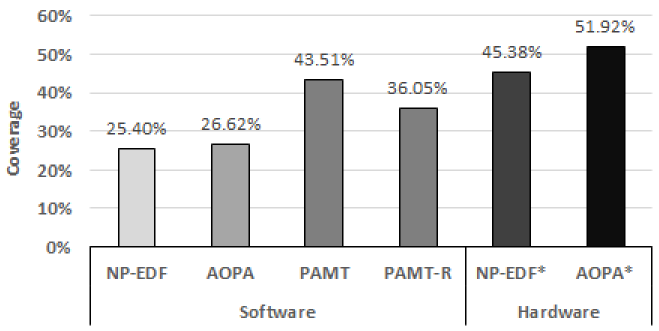

Fig. 14. Coverage of each priority assignment algorithm for the generated frame sets.

Simulation Configuration: 500Kbps is used as the bit-rate for the arbitration phase and 2Mbps for the data phase, because 500Kbps is commonly used for the powertrain network [13] and up to 2Mbps is supported by the current commercial CAN-FD transceiver which satisfies the automotive OEM's EMC requirement [14]. Also, we use 200μs as the mode-transition overhead according to our experimental measurement.

In addition, we use 11-bit IDs to simulate AOPA and AOPA* because there are typically about 100 different frames[3] for a single in-vehicle CAN bus [24]. and 11-bit ID suffices for that number. However, we use 29-bit IDs to simulate PAMT, PAMT-R, NP-EDF, and NP-EDF* because these frame-instance-level priority assignment algorithms require multiple IDs for a frame and 11 bits are not enough.

B. Results and Analysis

Coverage: is defined as the percentage of given frame sets whose *schedulable* priority order is found by each priority assignment algorithm. We evaluate the coverage of each priority assignment algorithm for the generated mixed frame sets. Fig. 14 plots the simulation results. As expected, PAMT outperforms the existing optimal priority assignments when the software solution is used. PAMT can find a schedulable priority order for 17–18% more mixed frame sets than the existing optimal priority assignments when we use the software solution. This is because PAMT effectively reduces the negative impact of mode transitions on schedulability by performing type-based clustering. Thus, for 17–18% additional mixed frame sets, we can use the economic software solution with PAMT. PAMT-R has 0.8% less coverage than PAMT because each frame has the reduced deadline. Also, the software solution is shown to have lower coverage than the hardware solution because the latter is not affected by the negative impact of mode transitions on schedulability. The coverage difference between PAMT and AOPA* (the maximum achievable coverage using 11-bit ID) is about 8%. Also, the coverage difference between PAMT and NP-EDF*

[3]We observed 96 different messages in 2015 Chevorlet Trax LT AWD through the On Board Diagnostic (OBD) port.

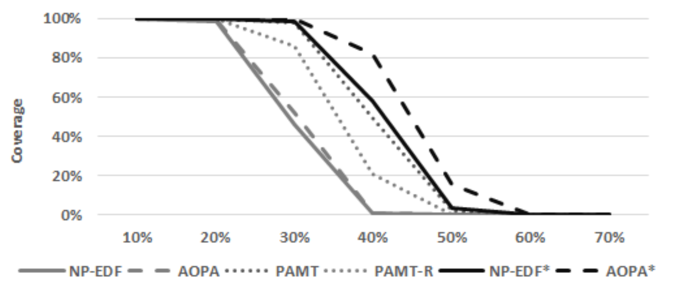

Fig. 15. Coverage of each priority assignment algorithm while varying utilization.

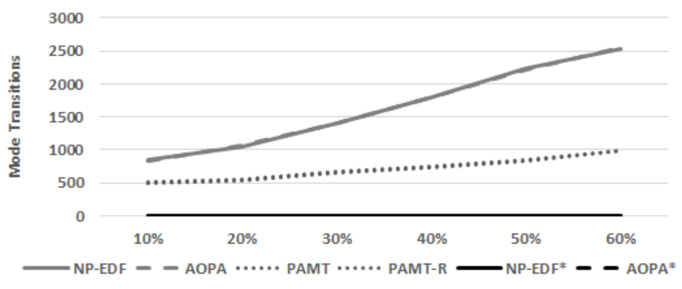

Fig. 16. The number of mode transitions required by each priority-assignment algorithm in a planning cycle (X-axis: utilization)

(the maximum achievable coverage using 29-bit ID) is about 2%. Because 29-bit ID is common in trucks [32], PAMT is now more useful for trucks, although this may change in future. Interestingly, the coverage of NP-EDF is lower than the coverage of AOPA due to its use of 29-bit ID.

Varying utilization: Fig. 15 shows the coverage of each priority assignment algorithm while varying the utilization by the generated frame set. PAMT outperforms the existing priority assignment algorithms regardless of the utilization by the mixed frame set when we use the software solution. In particular, Fig. 15 shows that PAMT can cover about over 97% of mixed frame sets if the utilization by the mixed frame sets is in the range of 30–40%. The number is 52% higher than NP-EDF and 46% higher than AOPA. The number of mode transitions incurred by each priority assignment algorithm is shown in Fig. 16. The number of mode transitions incurred by PAMT is much less than those incurred by AOPA and NP-EDF and the gap in the number of mode transitions between PAMT and the others becomes larger as the utilization of mixed frame sets increases. For example, the number of mode transitions incurred by AOPA is 1.6x larger than that by PAMT in the range of 10–20% and the gap becomes 2.5x in the range of 60–70%. Thus, the coverage improvement of PAMT over existing optimal priority assignments comes from the reduced number of mode transitions.

IX. Related Work

Priority assignment impacts greatly the schedulability of a given CAN frame set [11]. Representative frame level fixed priority assignment algorithms are Deadline minus Jitter Monotonic Priority Order (DJMPO) [36] and AOPA [5]. AOPA is proven to be optimal [26] if there is no priority inversion which could occur for various practical reasons [9, 17, 16]. Since frame-level fixed priority is less efficient than frame-instance-level fixed priority in utilization, use of the frame-instance-level fixed priority has been proposed [22, 35]. The representative frame-instance-level fixed priority assignment algorithm is NP-EDF, which is proven to be optimal among work-conserving scheduling algorithms for periodic tasks [15].

As in a typical electronic system, signals on CAN are interfered with by EMI [27], which may induce bit errors by distorting the signals. Since error recovery delays the delivery of CAN data frames, it impacts the schedulability of a given CAN frame set. Thus, Davis *et al.* [10] proposed a robust priority assignment algorithm which not only is optimal but also maximizes the number of successive tolerable transmission errors.

Since assigned IDs of the existing frame sets usually do not change even though new frames are introduced for new functions (e.g., updating an ECU), researchers focused on the backward compatibility of priority assignment. Schmidt [31] proposed a robust priority-assignment algorithm for a frame set when some frame IDs are fixed. Davis *et al.* [11] showed the existence of flaws in [31] when there is not an enough gap between fixed IDs and proposed a correction of robust priority assignment. Davis *et al.* [12] also consider optimal priority assignment under mixed use of FIFO queues and priority queues.

Previous studies of priority assignment for CAN focused on the single-type frame set, and thus they are agnostic of mode-transition overhead which must only be accounted for mixed frame sets. Therefore, the priority order determined by existing priority-assignment algorithms may incur many mode transitions. On the other hand, PAMT is aware of the mode-transition overhead, and clusters frame instances based on their type to minimize the mode-transition overhead.

X. Conclusion

Utilizing the silent mode of the existing CAN controller is a simple solution to solve the CAN and CAN-FD coexistence problem. However, it is non-trivial to minimize the negative impact of mode-transition overhead of the silent mode. To minimize the negative impact, we have proposed a new priority-assignment algorithm, called PAMT. PAMT minimizes the negative impact of mode transitions by clustering the frame instances based on their type, and is shown to be the optimal priority assignment for a mixed frame set. Also, our extensive simulation results show that PAMT outperforms existing priority-assignment algorithms in minimizing the negative impact of mode-transition overhead.

References

[1] http://www.worldometers.info/cars/.

[2] IEC 61508. Functional safety of electric / electronic / programmable electronic safety-related systems, 2011.

[3] T. Adamson. Hybridization of CAN and CAN FD networks. In *15th International CAN Conference*, 2015.

[4] Arduino. https://www.arduino.cc/.

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

[6] 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.

[7] Christelle Braun, Lionel Havet, and Nicolas Navet. NETCARBENCH: A benchmark for techniques and tools used in the design of automotive communication systems. In *7th IFAC International Conference on Fieldbuses & Networks in Industrial & Embedded Systems - FeT'2007*, pages 321–328, Toulouse, France, November 2007.

[8] 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.

[9] 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.

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

[11] 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.

[12] Robert I. Davis, Steffen Kollmann, Victor Pollex, and Frank Slomka. Schedulability Analysis for Controller Area Network (CAN) with FIFO Queues Priority Queues and Gateways. *Real-Time Syst.*, 49(1):73–116, January 2013.

[13] Eude Cezar de Oliveira. Electrical Architectures and In-Vehicles Networks. In *SAE Technical Paper*. SAE International, 04 2007.

[14] F. Hartwich. Bit Time Requirements for CAN FD. In *14th International CAN Conference*, 2013.

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

[16] 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.

[17] 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.

[18] D. A. Khan, R. I. Davis, and N. Navet. Schedulability analysis of CAN with non-abortable transmission requests. In *ETFA2011*, pages 1–8, Sept 2011.

[19] K. Lennartsson. CAN FD filter for Classical CAN controllers. In *15th International CAN Conference*, 2015.

[20] Microchip. *Stand-Alone CAN Controller with SPI Interface*, August 2012. Rev. G.

[21] S. Monroe, D. Stout, and J. Griffith. Solutions of CAN and CAN FD in a mixed network topology. In *14th International CAN Conference*, 2013.

[22] M. Di Natale. Scheduling the CAN bus with earliest deadline techniques. In *Proceedings 21st IEEE Real-Time Systems Symposium*, pages 259–268, 2000.

[23] N. Navet, Y.-Q. Song, and F. Simonot. Worst-case Deadline Failure Probability in Real-time Applications Distributed over Controller Area Network. *J. Syst. Archit.*, 46(7):607–617, April 2000.

[24] 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, pages 45–54, New York, NY, USA, 2016. ACM.

[25] Florian Pölzlbauer, Robert I. Davis, and Iain Bate. Analysis and Optimization of Message Acceptance Filter Configurations for Controller Area Network (CAN). In *Proceedings of the 25th International Conference on Real-Time Networks and Systems*, RTNS '17, pages 247–256, New York, NY, USA, 2017. ACM.

[26] 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.

[27] F. Ren, Y. R. Zheng, M. Zawodniok, and J. Sarangapani. Effects of Electromagnetic Interference on Control Area Network Performance. In *Region 5 Technical Conference, 2007 IEEE*, pages 199–204, April 2007.

[28] Robert Bosch GmbH. *CAN Specification*, 1991. Ver 2.0.

[29] Robert Bosch GmbH. *CAN with Flexible Data-Rate Specificaton*, April 2012. Ver 1.0.

[30] G. Rodriguez-Navas, S. Roca, and J. Proenza. Orthogonal, Fault-Tolerant, and High-Precision Clock Synchronization for the Controller Area Network. *IEEE Transactions on Industrial Informatics*, 4(2):92–101, May 2008.

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

[32] M. Schreiner, H. Mahmoud, M. Huber, S. Koc, and J. Waldmann. Safe-guarding CAN FD for applications in trucks. *CAN Newsletter*, 01/2013.

[33] M. Schreiner, H. Mahmoud, M. Huber, S. Koc, and J. Waldmann. CAN FD from an OEM point of view. In *14th International CAN Conference*, 2013.

[34] J. W. Shin, J. H. Oh, S. M. Lee, and S. E. Lee. CAN FD controller for in-vehicle system. In *2016 International SoC Design Conference (ISOCC)*, pages 227–228, Oct 2016.

[35] K. M. Zuberi and K. G. Shin. Non-preemptive scheduling of messages on controller area network for real-time control applications. In *Proceedings Real-Time Technology and Applications Symposium*, pages 240–249, May 1995.

[36] A. Zuhily and A. Burns. Optimality of (D-J)-monotonic priority assignment. *Information Processing Letters*, page 103(6), 2007.

XI. Appendix

A. Pseudocodes of PAMT Implementation

```
Algorithm 1: PAMT
Input : $I$: Mixed frame instance set
        $o_{mode}$: Mode transition overhead
Output: $I_{pas}$: Priority-assigned frame instance set

$t_v \leftarrow 0$; // Virtual time
$I_{cs} \leftarrow \{\}$; // Competing set
$I_{pas} \leftarrow \{\}$; // Priority assigned set
$N \leftarrow |I|$;
while $N \neq |I_{pas}|$ do
    $isMigrated \leftarrow false$;
    sort($I$); // By arrival time — ascending order
    for $k \leftarrow 0$ to $|I| - 1$ do
        if $I[k].arrival\_time \leq t_v$ then
            $I_{cs}[|I_{cs}|] \leftarrow I[k]$;
            $I[k] \leftarrow$ null;
            $isMigrated \leftarrow true$;
        end
    end
    if $isMigrated = false$ and $|I_{cs}| = 0$ then
        $t_v \leftarrow I[0].arrival\_time$;
        continue;
    end
    sort($I_{cs}$); // By deadline — ascending order
    $f_{sel} \leftarrow I_{cs}[0]$; // $f_{sel}$: selected frame instance by NP-EDF
    $idx_{sel} \leftarrow 0$;
    for $k \leftarrow 1$ to $|I_{cs}| - 1$ do
        if $I_{cs}[k].deadline = f_{sel}.deadline$ and
          $I_{pas}[|I_{pas}| - 1].type = I_{cs}[k].type$ then
            $f_{sel} \leftarrow I_{cs}[k]$;
            $idx_{sel} \leftarrow k$;
        end
    end
    $I_{pas}[|I_{pas}|] \leftarrow f_{sel}$;
    $I_{cs}[idx_{sel}] \leftarrow$ null;
    if $Cluster(I_{pas}) = true$ then
        $t_v \leftarrow t_v + f_{sel}.transmission\_time$;
    end
    else
        $t_v \leftarrow t_v + f_{sel}.transmission\_time$;
        if $I_{pas}[|I_{pas}| - 1].type \neq I_{pas}[|I_{pas}| - 2].type$
          then
            $t_v \leftarrow t_v + o_{mode}$;
        end
        $I_{pas}[|I_{pas}| - 1].completion\_time \leftarrow t_v$;
        if $I_{pas}[|I_{pas}| - 1].completion\_time >$
          $I_{pas}[|I_{pas}| - 1].deadline$ then
            return null; // Declare unschedulable
        end
    end
end
return $I_{pas}$
```

```
Algorithm 2: Cluster
Input : $I_{pas}$: Priority-assigned frame instance set

$i \leftarrow |I_{pas}| - 1$;
// Check C1
if $I_{pas}[i].type = I_{pas}[i-1].type$ then
    return false; // Already clustered
end
$isSameTypeExist \leftarrow false$;
$k \leftarrow 0$;
for $j \leftarrow 2$ to $|I_{pas}| + 1$ do
    if $I_{pas}[i].type = I_{pas}[i-j].type$ then
        $isSameTypeExist \leftarrow true$;
        $k \leftarrow j$;
        break;
    end
end
if $isSameTypeExist = false$ then
    return false;
end
// Check C3
if $I_{pas}[i].arrival\_time > I_{pas}[i-k+1].arrival\_time$
  and $I_{pas}[i].arrival\_time > I_{pas}[i-k].completion\_time$
  then
    return false;
end
// Check C2
for $p \leftarrow k-1$ to $0$ do
    if $I_{pas}[i-p].completion\_time +$
      $I_{pas}[i].transmission\_time > I_{pas}[i-p].deadline$
      then
        return false;
    end
end
$temp \leftarrow I_{pas}[i]$;
for $p \leftarrow 0$ to $k-1$ do
    $I_{pas}[i-p] \leftarrow I_{pas}[i-p-1]$;
    $I_{pas}[i-p].completion\_time \leftarrow I_{pas}[i-$
      $p].completion\_time + temp.transmission\_time$;
end
$I_{pas}[i-k+1] \leftarrow temp$;
$I_{pas}[i-k+1].completion\_time \leftarrow$
  $I_{pas}[i-k].completion\_time + temp.transmission\_time$;
return true;
```