The Frame Packing Problem for CAN-FD

Unmesh D. Bordoloi
Linköping University, Sweden
unmesh.bordoloi@liu.se

Soheil Samii
General Motors
soheil.samii@gm.com

***Abstract*—CAN with flexible data rate (CAN-FD) allows transmission of larger payloads compared to standard CAN. However, efficient utilization of CAN-FD bandwidth space calls for a systematic strategy. The challenge arises from the nature of the frame sizes stipulated by CAN-FD as well as the heterogeneity of the periods of the messages and the signals. In this paper, we formulate a frame packing problem for CAN-FD with the optimization objective of bandwidth utilization while meeting temporal constraints. As part of the solution, first, we propose a formula to compute the best-case and the worst-case transmission times of the CAN-FD frames. Thereafter, we propose a framework that solves the optimization problem in pseudo-polynomial time. Experiments show the gains achieved by our framework. The results also show that, when applied to standard CAN, our heuristic provides improved results over existing techniques.**

I. Introduction

CAN with Flexible Data-rate (CAN-FD) [10], [7] is currently being added to the CAN standard (ISO 11898) and is the next-generation CAN technology to be used for in-vehicle networks and industrial automation [4]. The main benefit of CAN-FD compared to CAN 2.0 is the significant increase in bus bandwidth. This improvement is enabled essentially by two enhancements to data-link layer standard of CAN: (1) The data-bit rate is increased (up to 8 Mbps is envisioned although this depends on physical properties such as wire lengths, distance between ECUs, and electromagnetic interference) and (2) a new frame format is introduced to allow frame payloads of up to 64 bytes—thus increasing bandwidth efficiency on a per-frame basis. As a point of comparison, current CAN technology supports a maximum payload size of 8 bytes and is usually operated at 500 kbps.

Although the data-bit rate is increased, the inherent arbitration phase in CAN remains at an unchanged bit rate (i.e., 500 kbps for most systems). Arbitration is a part of every frame transmission and comprises a fixed portion of the total frame transmission time independent of the size of the frame payload. This portion is very significant for short frames (e.g., 8-byte frames) but is decreased for frames with large payloads, such as the ones available in CAN-FD. For this reason, frames with larger payload should be used to achieve efficient bus usage. Messages and signals produced by sensors and Electronic Control Units (ECUs) are typically transmitted at different rates and are of sizes that are significantly shorter than the large payloads in CAN-FD. To use large payloads in CAN-FD efficiently, it is therefore needed to pack multiple messages and signals in one single frame. The frame packing problem is complex due to the variation in the size, periodicity, and time criticality of the individual signals. The stipulated frame sizes in CAN-FD (Section III) introduce further complexity in the frame packing problem.

Contribution: In this paper, we motivate and solve a frame packing problem for CAN-FD systems, where the goal is to optimize the bus usage while satisfying the timing constraints imposed on individual signals. Frame packing is an enabler to use CAN-FD buses efficiently in terms of bus usage. The actual frame-packing strategy and algorithm to map individual signals to frames not only has an impact on the amount of data that can be transmitted on the bus, but also the bandwidth margin for future feature growth during the lifetime of the CAN-FD subsystem. Frame packing also reduces or eliminates the need to add additional buses to the architecture to enable feature expansion.[1]

Towards an efficient and effective frame-packing strategy, we propose an optimization approach to select and pack messages into frames to minimize bus utilization. Further, we present an approach to assign frame identifiers (i.e., priorities) to the constructed frames, while considering timing constraints and potential re-packing of messages to satisfy individual deadlines. We also provide new formula required to compute best-case and worst-case transmission times of CAN-FD frames in light of the new frame format, increased bit rate, and changes in the bit stuffing rules. Experiments show that our approach is essential to find system configurations with efficient usage of the increased communication bandwidth offered by CAN-FD. In addition, the results demonstrate that our heuristic—when applied to standard CAN—improves over known techniques.

Organization: The next section surveys related work in the area of frame packing and bin packing heuristics. Section III gives a technical overview of the CAN-FD protocol, focusing in particular on the differences to standard CAN, as well as calculation of best and worst case transmission times of individual CAN-FD frames. In Section IV, we present comprehensive examples to demonstrate the challenges and advantages of CAN-FD frame packing. Section V presents the formal problem statement and system model. Our optimization approach, comprising frame packing and priority assignment, is presented in Section VI. Experimental results are presented in Section VII and the paper is concluded in Section VIII.

II. Related Work

This is the first paper to address the problem of frame packing for CAN-FD systems. Prior to our paper, there have been a few papers in the literature that proposed frame packing scheme for standard CAN [11], [13], [14].

[1] Adding an additional CAN bus to an existing architecture comes with significant costs and problems in terms of additional transceivers and CAN controllers, change requests involving increased interaction with multiple suppliers, development and validation effort, wiring complexity and weight, and deviations from system-level architectural decisions and directions because ECUs originally not intended to act as gateways between different subnets now must gate frames between multiple CAN subnets.

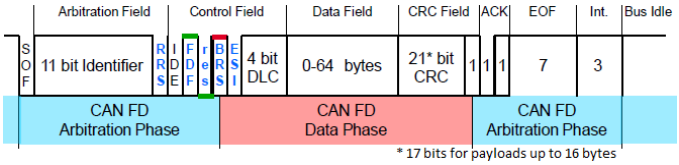

Fig. 1. CAN-FD frame format [7]. A frame includes a frame identifier, which can be used to incorporate transmission priorities as well as to identify the content of the payload. The BRS bit is used to switch to a higher bit rate for data transmission of up to 64 bytes data.

Sandstrom et al. [14] proposed a heuristic inspired by the next fit decreasing heuristic for the classic bin packing problem [16]. Their algorithm sorts the signals by deadlines and then iteratively assign a signal to a frame until a signal does not fit into the frame anymore, whereafter a new frame is created for the next iteration. Saket and Navet [13], on the other hand, sort the signals by their bandwidth utilization. Thereafter, the sorted list is processed alternatively from the beginning and from the end to increase the chances that signals with similar periods are packed in the same frame. Finally, Polzlbauer et al. [11] presented a heuristic based on the next fit decreasing algorithm, which improves on the previous approaches. Compared to previous techniques, they proposed an additional criteria that decides whether to add a signal to an existing frame based on the impact on bandwidth utilization.

Standard CAN allows payloads with 0, 1, 2, ..., 8 bytes. As such, the bandwidth loss within a frame is relatively less and when the sizes of signals to be packed into a frame are specified in bytes, there is no bandwidth loss at all. This is different in CAN-FD because the relative bandwidth loss may become significant due the discontinuity in the allowed frame sizes (in addition to the standard CAN sizes, CAN-FD allows payloads of 12, 16, 20, 24, 32, 48, or 64 bytes). Under such conditions, the problem discussed in this paper does not arise in the context of standard CAN and has not been discussed in related papers such as the ones discussed in this section. It is, however, possible to apply our approach to standard CAN and it performs better than existing solutions in literature (Section VII-D).

Other notable work on frame packing include papers that have targeted communication protocols like FlexRay [15] and mixed event/time-triggered networks [12] and, as such, the nature of those frame packing problems are very different.

The problem addressed in this paper has some similarities with the *variable sized bin packing problem*. There are important differences, however, that are discussed in Section V-D. While the classic bin packing problem has been widely studied and excellent resources are abundant in the literature, the variable sized bin packing has received relatively limited attention. The variable sized bin packing problem was studied for the first time by Friesen and Langston [5] who proposed three heuristics. Murgolo [9] provided theoretical results on polynomial-time approximation schemes. Another variant of the problem studied in the literature is known as the variable sized bin packing problem with *fixed costs*, where a fixed cost is associated with each bin and the objective is to minimize the total costs [8], [1]. Note that our frame packing problem has variable costs that depend on the periods of the selected frames.

III. CAN-FD Overview

We shall in this section give an overview of the CAN-FD protocol and in particular highlight the differences to the standard CAN protocol. We shall also show how to calculate worst-case and best-case transmission times of CAN-FD frames.

A. Frame Format and Operation

Figure 1 depicts the frame format in CAN-FD networks. When the bus is idle, the nodes that have a frame to send first transmit a dominant (logical 0) bit, denoted SOF (Start-Of-Frame), followed by an 11-bit identifier in the arbitration phase. This transmission is done in the arbitration bit rate, where each bit has a duration denoted t_{arb} (for a setup where the arbitration bit rate is 500 kbps, the bit time t_{arb} is 2 μs). Whenever, during the transmission of the 11 identifier bits, a node transmits a recessive bit (logical 1) and notices a dominant value (logical 0) on the bus, it stops transmitting (because at least one other node is going to transmit a higher-priority frame) and instead prepares to receive a frame being transmitted by another node. After the transmission of the 11 identifier bits is completed, one node is the transmitter and the other nodes will act as receivers. The transmitting node then continues by transmitting the bits in the control field (the RRS bit at the end of the arbitration field is 0 and is reserved for future protocol extensions). The IDE bit indicates whether the frame uses additional identifier bits (extended frame format). In essence, the above discussed the responsible bits (and thereby, the time taken) for the arbitration in CAN-FD and the arbitration policy itself, which is based on priorities.

The FDF bit (FD Format) is 1 if the frame is encoded with the CAN-FD frame format and 0 if the frame is encoded according to the standard CAN 2.0 format. We consider FDF to be 1 for all frames (the old format is available for backwards-compatibility reasons). The following bit (res) is reserved for future protocol variants.

The BRS bit (Bit-Rate Switch) is 1 if the bus controllers should switch to the increased data rate, where the duration t_{data} of each bit is configured[2] to be significantly shorter than the arbitration bit time t_{arb}. We consider in this paper that BRS is 1 for all frames, thus taking advantage of the increased bit rate offered by the CAN-FD protocol. We believe that there are interesting open problems related to selection of the BRS bit but they are not the subject of this paper. [3]

ESI (Error Status Indicator) indicates whether the transmitting node is in error passive or error active state. This ESI bit is—with the introduction of CAN-FD—a dependability-related enhancement to the CAN standard and is not relevant for the topic of this paper. Following the ESI bit is a 4-bit data-length code (DLC). The DLC indicates the number of bytes in the following payload section. The available payload sizes are 0 to 8 bytes, as well as 12, 16, 20, 24, 32, 48, and 64 bytes (CAN 2.0 allows a

[2]The configuration of the bit times t_{arb} and t_{data} is static for a CAN-FD network and depends on factors such as the number of nodes, length of wires, physical distance between nodes, and availability of transceivers qualified for the intended bit rate.

[3]Reasons to restrict operation to the arbitration bit rate for the whole transmission (i.e., BRS = 0) may be due to legacy applications or to reduce the error probability in the physical layer.

maximum size of 8 bytes). Following the payload is a CRC (Cyclic Redundancy Check) field, which consists of 17 bits for payloads up to 16 bytes, and 21 bits for payloads larger than 16 bytes. The next bit is the CRC delimiter, which is transmitted as a recessive bit (1). The bit time is changed during the bit time for the CRC delimiter, which means that the remaining sections ACK (Acknowledgment), EOF (End Of Frame), and Int (inter-frame gap) are transmitted with the arbitration bit rate.

To summarize, the main differences to CAN 2.0 are as follows:

- The three bits FDF (FD Format), BRS (Bit Rate Switch), and ESI (Error Status Indicator) are introduced.
- A longer payload size is supported. The allowed payload size in a CAN 2.0 frame range from 0 to 8 bytes, whereas, for a CAN-FD frame, it is possible to additionally transmit frames with 12, 16, 20, 24, 32, 48, and 64 bytes payload. The larger available payloads introduce opportunities for bandwidth optimizations by considering multiple messages or signals in one single frame (the topic of this paper).
- After arbitration has been completed, it is possible to switch to an alternate faster bit rate. By setting FDF to 1, the bit time is switched from the arbitration bit time t_{arb} to the shorter data bit time t_{data}.

B. Bit Stuffing

The CAN protocol has a bit stuffing rule, which means that if five bits are transmitted consecutively as either dominant or recessive, then the communication controller transmits one stuff bit that has the opposite value than the consecutive bits. This mechanism is used for the receivers to detect errors on the bus. The bit stuffing mechanism is included in CAN-FD as well, with minor changes: The bit-stuffing rule is only applied from the start of a CAN-FD frame to the end of the payload section. For the CRC field, stuff bits are static and independent of the actual CRC value. For a 17-bit CRC, 4 stuff bits are added (making the CRC to be 21 bits in total), whereas for a 21-bit CRC, 5 stuff bits are added (making a total size of 26 bits).[4] Given these differences, previously proposed formula [2] to compute the transmission times are not directly applicable to CAN-FD.

C. Transmission-Time Calculation

The best-case transmission time occurs when no stuff bits are inserted during frame transmission. In such a scenario, the number of bits transmitted with the arbitration bit rate is 29 (including the BRS bit and excluding the CRC delimiter bit[5]). The number of bits transmitted in the data bit rate depends on the size of the CRC field (which depends on whether or not the payload is larger than 16 bytes). For payloads up to 16 bytes, the CRC field is 21 bits (17 bits CRC and 4 mandatory stuff bits). For payloads larger than 16 bytes, the CRC field is 26 bits (21 bits CRC and 5 mandatory stuff bits). Considering this property and counting the bits that are transmitted in the data bit rate (excluding BRS and including the CRC delimiter), we conclude that the best-case transmission time of a CAN-FD frame with p bytes payload is

$$\mathsf{BCTT}(p) = 29t_{\text{arb}} + \left(27 + 5\left\lceil\frac{p-16}{64}\right\rceil + 8p\right)t_{\text{data}}. \quad (1)$$

For the worst-case transmission time, we consider a worst-case bit stuffing scenario. The bit stuffing rule applies until the end of the data field. Let us consider the bits transmitted with the arbitration bit rate before the first bit rate switch: Since the FDF, res, and BRS bits are 1, 0, and 1, respectively, there are 13 bits that are subject to the bit stuffing rule. Considering a worst-case bit pattern of 11111000011110000 . . . [2], we note that there are at most 3 stuff bits transmitted with the arbitration bit rate. Let us further consider the bits transmitted with the data bit rate: The bits that are subject to bit stuffing are the DLC and payload (data field), which is a total of $4+4p$ bits for a p bytes payload. Considering that BRS is 1 and that ESI may also be transmitted as 1, we need to consider these two additional bits in the worst-case bit sequence. Thus, the maximum number of stuff bits transmitted in the data bit rate is $\lfloor(6+8p)/4\rfloor = 1+2p$. Adding the contribution of the worst-case bit stuffing patterns to Equation 1, we conclude that the worst-case transmission time of a CAN-FD frame with p bytes payload is

$$\mathsf{WCTT}(p) = 32t_{\text{arb}} + \left(28 + 5\left\lceil\frac{p-16}{64}\right\rceil + 10p\right)t_{\text{data}}, \quad (2)$$

where $p = 0, 1, \ldots, 8$ or $p = 12, 16, 20, 24, 32, 48, 64$ (corresponding to the available payload sizes in a CAN-FD frame). Equations 1 and 2 are valid for the case that the BRS bit is transmitted as a logical 1 (recessive bit). In case a frame is transmitted without taking advantage of the increased bit rate (i.e., BRS is 0), the transmission time is calculated by substituting t_{data} with t_{arb} in Equations 2 and 1. We shall in this paper consider that BRS is 1 for all frames.

To illustrate the different components of the worst-case transmission time, let us consider Figure 2. The chart shows, for each available CAN-FD payload size, four

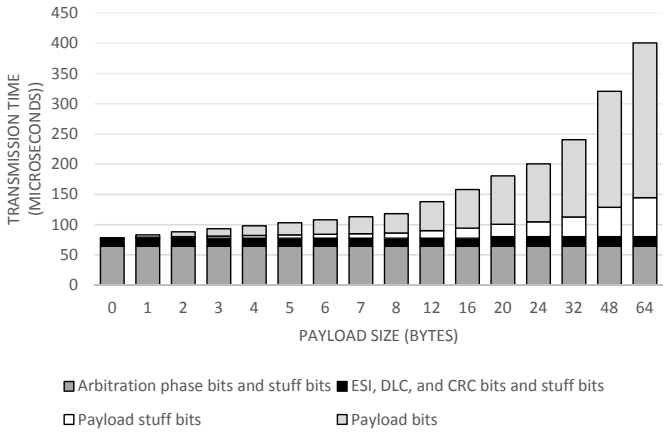

Fig. 2. Worst-case transmission time (μs) as a function of CAN-FD payload size (bytes). The graph shows, for each available payload size, the total worst-case transmission time consisting of four time components with the following order from below: (1) Bits transmitted with arbitration bit rate, (2) ESI, DLC, and CRC bits including stuff bits, (3) payload stuff bits, and (4) payload bits.

[4]For CAN 2.0, the CRC field follows the same bit stuffing rule as the preceding fields.

[5]The two bit rate switches in a CAN-FD frame transmission are actually performed *during* the transmission of the BRS bit and CRC delimiter bit, respectively.

time components that together constitute the worst-case transmission time of a CAN-FD frame, considering 500 kbps arbitration bit rate (i.e., $t_{\text{arb}} = 2\mu$s) and 2 Mbps data bit rate (i.e., $t_{\text{data}} = 0.5\mu$s). The dark-gray field (first from below) is the time to transmit all bits in the arbitration bit rate (i.e., $32t_{\text{arb}}$). The black field (second from below) is the time to transmit the ESI bit, the DLC section, and the CRC field including the mandatory stuff bits. The white field (third from below) shows the time to transmit only the stuff bits that in the worst-case are inserted into the DLC and payload sections (i.e., $(1 + 2p)t_{\text{data}}$, where p is the payload size in bytes). Finally, the light-gray field (the fourth and last from below) is the time spent to transmit the actual payload (i.e., $8pt_{\text{data}}$). Note that the first three time components together show the time-overhead of the CAN-FD protocol in transmitting data of a certain size. We can observe that the overhead is dominating for small CAN-FD frames and that the communication medium is best utilized when transmitting large CAN-FD frames.

IV. Motivational Examples

Despite the above observation that favors larger frame sizes, packing frames for CAN-FD bus with optimal bus usage is a nontrivial problem. In this section, we illustrate this through a series of examples. We shall show different alternatives to pack signals into frames, and also situations for which frame packing is not optimal. We shall also consider to pack signals with different periods into the same frame to further optimize bus usage. Unless otherwise stated, we shall consider a CAN-FD network with the following configuration: $t_{\text{arb}} = 2\mu$s and $t_{\text{data}} = 0.5\mu$s.

a) Frame packing to optimize bus utilization: Let us consider three signals s_1, s_2, and s_3 produced by a single ECU with the sizes $l_1 = 4$, $l_2 = 2$, and $l_3 = 8$ bytes, respectively. All three signals are produced with a period of 10 ms. First, consider that each signal is transmitted in a separate frame. The total worst-case transmission time is $\mathsf{WCTT}(4) + \mathsf{WCTT}(2) + \mathsf{WCTT}(8) = 304\mu$s, which gives a bus utilization of $304/10000 = 3.04\%$.

To continue the example, consider that we pack the signals into frames such that each frame is fully utilized. One such example is to pack s_1 and s_2 into a 6-byte frame, and keep s_3 in a single 8-byte frame. This leads to a total worst-case transmission time of $\mathsf{WCTT}(6) + \mathsf{WCTT}(8) = 226\mu$s and a bus utilization of 2.26%.

To further optimize the bus usage, we can consider another alternative packing to combine all three signals into a single CAN-FD frame. The total length of the three signals is 14 bytes, which means that a 16-byte frame is needed (the available CAN-FD payload length is not continuous after 8 bytes). This means that 2 unnecessary bytes will be transmitted, but this slight wastage of bus utilization is smaller than the bus utilization that is reduced when combining all signals into one single frame. Considering $\mathsf{WCTT}(16) = 158\mu$s, the bus utilization for this example is 1.58%.

The examples show that alternative ways of packing influence the bus bandwidth consumption. Further, we demonstrated that it is not straightforward to identify which packing alternative is the best in terms of bus utilization. For large systems, the design space is very complex due to the many combinations that are possible. Note that there is a bus usage overhead associated with each frame, apart from its actual payload, due to for example the frame identifier, DLC, and CRC fields (refer back to Figure 2). If each signal is sent in a separate frame, then this overhead is associated with each signal. If, on the other hand, signals are packed together into one frame, then the overhead is relatively smaller because the overhead is associated to the frame and not the individual signals. This is captured in the transmission-time computation in Equation 2 and is accounted for by our optimization approach in Section VI.

b) Frame packing of signals with different periods: To further demonstrate the benefits of frame packing, let us consider an example with three signals s_1, s_2, and s_3, each with 4 bytes of data. The periods of the three signals are $h_1 = 10$ ms and $h_2 = h_3 = 20$ ms. If each signal is transmitted in a separate frame (i.e., no frame packing is considered), the bus utilization is

$$\sum_{i=1}^{3} \frac{\mathsf{WCTT}(4)}{h_i} = \frac{98\ \mu\text{s}}{10\ \text{ms}} + \frac{98\ \mu\text{s}}{20\ \text{ms}} + \frac{98\ \mu\text{s}}{20\ \text{ms}} = 1.96\%.$$

Let us now consider that we perform frame packing but with the restriction that signals with different periods are not mixed within a frame: Signal s_1 is transmitted in a separate frame with the period 10 ms, whereas signals s_2 and s_3 are packed together in an 8-byte frame that is transmitted with the period 20 ms. This results in a bus utilization of

$$\frac{\mathsf{WCTT}(4)}{10\ \text{ms}} + \frac{\mathsf{WCTT}(8)}{20\ \text{ms}} = 1.57\%.$$

Finally, let us pack all three signals into a single 12-byte CAN-FD frame. This frame is transmitted with a period of 10 ms, because of the period of s_1. Because the period of s_2 and s_3 is 20 ms, for any two continuous transmissions of the 12-byte frame, one transmission will only include s_1 and leave the remaining 8 bytes of payload with no useful data. For every other transmission, we will thus waste the amount of bandwidth it requires to transmit 8 bytes of data with the data bit rate. However, we reduce the bus utilization compared to the previous packing alternative because we reduce the solution by one CAN-FD frame (a CAN-FD frame comes with a certain overhead due to the transmission of the frame header regardless of the payload size). The bus utilization is reduced to $\mathsf{WCTT}(12)/10$ ms $= 1.38\%$. This shows that it can be beneficial to combine multiple signals with different periods into a single frame in order to further optimize bus utilization, even if some frames will be larger than necessary (i.e., bandwidth is wasted for certain frame transmissions). This can be compared to the case where the bus utilization is 1.57%. In that case, all frames are packed to their limit but signals with different periods are packed in separate frames. Thus, considering signals with different periods makes the frame packing more complex but also presents opportunities for further optimizations.

c) Frame packing may increase bus utilization: Let us in this example consider a configuration where $t_{\text{arb}} = t_{\text{data}} = 0.5\mu$s. Consider two signals s_1 and s_2 of the same size $l_1 = l_2 = 20$ bytes and same period. Further, consider that each of the two signals is transmitted in separate 20-byte CAN-FD frames. The total worst-case transmission time of these two frames is $2 \times \mathsf{WCTT}(20) = 1060\mu$s. Consider now the case where s_1 and s_2 are packed together

in a single frame. The smallest available CAN-FD frame size is 48 bytes, which means that we have to transmit 8 additional bytes in order to send the two signals in the same frame. This leads to a worst-case transmission time of $\mathsf{WCTT}(48) = 1090\mu$s. This shows that there are cases where it is beneficial to keep signals in separate frames. However, in practice, the data bit rate will be higher than the arbitration bit rate, which means that the 8 additional bytes that have to be transmitted will consume less time than the time it takes to transmit the frame header of an additional frame in the arbitration data rate. In any case, our approach considers the trade-off between packing and not packing signals into frames.

d) Schedulability: The examples in this section focused on optimizing bus utilization. For a given frame packing, the next step is to assign unique frame identifiers, which serve as transmission priorities at runtime. For a given assignment of frame identifiers, it is needed to run a timing analysis to check whether or not all signal deadlines are met. It may be the case that the frame packing that leads to the best bandwidth utilization is not schedulable in the sense that it is not possible to assign frame identifiers such that all deadlines are met. In such cases, other frame packing solutions need to be explored in order to find an optimized and schedulable frame packing—for example, by not combining signals with different time criticalities into the same frame. In addition to a frame packing optimization approach, this paper addresses the issue of schedulability in the context of frame packing.

V. Problem Formulation

As an input, we are given sets of signals from all ECUs (Electronic Control Units) as described in Section V-A. The output that must be provided is a set of frames, where each frame contains a subset of signals from one ECU. The output should also provide the period, deadline, size as well as the priority identifier for each frame as defined in Section V-B. The optimization objective is to minimize the bandwidth utilization while satisfying the constraint that all the frames meet their deadlines (Section V-C).

A. Input: Signals

As an input, we are given a set E of ECUs in a distributed automotive system connected to a single CAN-FD bus. The eth ECU produces a set of signals denoted by $\mathbf{S}^e = \{s_1^e, s_2^e, \cdots, s_{|\mathbf{S}|}^e\}$. Each signal s_i^e is characterized by the following parameters.

- **Period, h_i^e:** denotes the rate at which signal s_i^e is produced.
- **Deadline, d_i^e:** is the latest time instant, relative to the instant when the signal s_i^e is produced, by which the transmission of s_i^e must be completed.
- **Length, l_i^e:** denotes the size (payload) of the signal s_i^e in bits. If the signal is transmitted as a frame in itself, then it must be packed into the smallest legal CAN-FD frame size that accommodates it. Given the size of the signal l_i^e, let $LT(l_i^e)$ be a function (or a look-up table) that gives us the CAN-FD frame size for this signal. [6] The time taken to transmit the entire signal is $\mathsf{WCTT}(LT(l_i^e))$.

[6]For example, given 14 bytes (112 bits) as the input, the function LT will return 16 bytes which is the smallest legal CAN-FD frame size that accommodates 14 bytes.

B. Output: Frames

As output, the problem packs all the signals into a set of frames. A frame γ_j^e consists of a set of signals $\mathbf{S}_j^e$, where $\mathbf{S}_j^e \subseteq \mathbf{S}^e$. Note that signals from two different ECUs are not allowed to be packed into the same frame. Parameters of a frame are defined as follows.

While packing a group of signals into one frame γ_j^e, the period H_j^e (and the deadline D_j^e) of the resulting frame will be the minimum period (minimum deadline) among the periods (deadlines) of signals. To find the length L_j^e, of a frame γ_j^e, we compute the summation of the payload of the constituent signals $\sum_{i=1}^{|\mathbf{S}_j^e|} l_i^e$. Once the length is known, it is trivial to find the worst-case transmission time $\mathsf{WCTT}(LT(L_j^e))$, using Equation 2. As before, the function $LT(L_j^e)$ gives us the smallest legal frame size in CAN-FD that may accommodate the payload size L_j^e. It should be noted that a frame is also associated with a priority identifier ID_j^e that uniquely identifies the priority of the frame for purposes of arbitration.

C. Optimization Objective and Constraints

In the following, we discuss the optimization objective of the problem and the constraints that it must satisfy.

$$\text{Minimize} \sum_{e=1}^{|E|} \sum_{j=1}^{|\mathbf{S}|} \sum_{k=1}^{m} y_{jk}^e \left(\frac{\mathsf{WCTT}(W_k)}{H_{jk}^e} \right) \quad (3)$$

$$\text{Subject to} \sum_{i=1}^{|\mathbf{S}^e|} x_{ijk}^e l_i^e \leq y_{jk}^e W_k \text{ , for all } j,\ k \text{ for each } e \quad (4)$$

$$\sum_{j=1}^{|\mathbf{S}^e|} \sum_{k=1}^{m} x_{ijk}^e = 1 \text{ , for all } i, \text{ for each } e \quad (5)$$

Above, m denotes the number of types of frames allowed. For CAN-FD $m = 16$ because the available payload sizes are 0 to 8 bytes, as well as 12, 16, 20, 24, 32, 48, and 64 bytes. The size of the frame of type k is given as W_k and its worst-case transmission time is then given as $\mathsf{WCTT}(W_k)$. It is trivial to observe that no solution may ever contain more frames for a given ECU than the given number of signals produced by that ECU. This means that the number of each type of frame in the solution is upper bounded by $|\mathbf{S}^e|$. Given this bound, y_{jk}^e is a boolean variable that refers to the selection of the jth frame out of $|\mathbf{S}^e|$ frames of eth ECU for frame of type k. Given these terms, it may be seen that Equation 3 is the objective function that minimizes overall bandwidth utilization considering the selected frames for each of the signals.

Let x_{ijk}^e be a boolean variable that denotes whether signal s_i^e from ECU e is selected for the jth frame of type k of ECU e. Equation 4, then, enforces the constraint that the sum of the payload of constituent signals of a single frame should not exceed the size of payload. Finally, Equation 5 enforces that each signal may be packed into one frame. Only the frames from same ECU may be packed together, and as such, the above two constraints must hold true for each ECU.

Let S_j^e be the subset of signals that are packed into the jth frame in ECU e (dropping the subscript k, i.e, regardless of the type of the frame). The period H_j^e for a

frame is determined by the constituent signals as shown in Equation 6.

$$\begin{cases} H_j^e = \min_{i=1}^{|\mathbf{S}_j^e|}\{h_i^e\} \\ D_j^e = \min_{i=1}^{|\mathbf{S}_j^e|}\{d_i^e\} \end{cases} \tag{6}$$

Finally, note also that the priorities (ID) must be assigned such that the response time of each frame must be less than its deadline. It should be noted that the priority assignment must consider signals from all ECUs that share the same CAN-FD bus.

$$R_j^e \leq D_j^e \tag{7}$$

The response time for frames may be computed using state-of-the-art techniques [2] used for standard CAN that remain valid for CAN-FD. However, the new formula to compute the worst-case transmission times should be used (Equation 2).

D. Hardness

The frame packing problem for CAN-FD formulated above is NP-hard and may be seen as a more complex variant of the *bin packing problem with variable sizes*. The bin packing problem with variable sizes considers that m types of bins are given and that a set of n items that must be packed into these bins. The optimization objective is to minimize the number of bins used. The similarities with the CAN-FD frame packing problem appear when one relates the bins with the CAN-FD frames sizes and the items with the signals.

Although there are similarities between the CAN-FD frame packing problem and the bin packing problem with variable sizes, there are important differences that increase the difficulty of the problem studied in this paper. The bin packing problem has the objective of minimizing either (i) the number of bins (frames in our setting) or, equivalently, (ii) the total bin size that is being wasted. For the frame packing problem, however, the goal is to minimize the bandwidth utilization because we have to account for the bandwidth wastage due to the varying periods of the signals in a frame as well (example in Section IV). As such, we have a different objective function (Equation 3) when compared to the classic bin packing problem. The variable frame period H_{jk}, which depends on the periods of the signals packed in that frame, leads to nonlinearities in Equations 3 and 6. Moreover, in the frame packing problem, we are not only interested in optimizing the bandwidth but also in the schedulability of the resulting frames (the classic bin packing problem has no such temporal constraints).

VI. Proposed Approach

An overview of our proposed optimization approach, for the problem outlined in Section V-C, is shown in Figure 3. Our framework consists of two major stages. The first stage produces a packing that is optimized from the point of view of bandwidth utilization. This optimization approach is inspired by algorithmic techniques that solve the bin-packing problem by invoking the subset sum problem in an iterative fashion [6].

The second stage begins by verifying the schedulability of the frames and is based on a well known algorithm [2], [3] and Equation 2. If the set of frames is not schedulable, the heuristic proceeds to find out the signal(s) that led to the deadline violation and proposes a new packing. Thereafter, the algorithm iterates until a feasible packing is produced or until it declares infeasibility.

It should be clarified that the above approach does not necessarily mandate the de-coupling of the overall problem into two separate problems — a packing problem followed by a scheduling problem. A feedback loop is integrated into stage two that allows us to search the design space for less optimized, but feasible solutions in case the first solution turns out to be unschedulable. More details about both stages follow.

A. Stage One: Optimized Packing

The first step of our heuristic is to find an optimized packing of the signals. As discussed in Section V-D, although this problem has similarities with the bin-packing problem, there are important differences. As such, classical approaches to solve the bin-packing problem may not be directly applied here. In light of this, while we propose a solution that is inspired by approaches that solve the classic variable sized bin-packing problem, the core algorithm that we design is completely different.

An approach to solve the variable sized bin-packing involves iteratively solving several subset sum problems [6]. At each iteration, the heuristic solves a subset-sum problem for each type of bin considering all the unpacked items. For each type of bin, this yields the set of items that lead to minimum waste. Among all the bins, the heuristic then chooses the bin with the best packing and removes the set of items that were packed into this bin. Thereafter, the heuristic proceeds to the next iteration and continues until all the items are packed.

Our heuristic works in a similar fashion, i.e., in each iteration, it chooses a new CAN-FD frame size and packs a subset of the signals into it. The iterations continue until all the signals have been packed. At each iteration, the *best* CAN-FD frame is selected. Towards this, unlike bin-packing, we cannot simply solve a subset-sum problem for each CAN-FD frame size. As discussed before (Section V-D), the complexity arises due to the nature of the objective function that must account for bandwidth utilization considering the periods. We define the problem to be solved in our case and refer to it as the ***CAN-FD Frame Selection*** (CaFeS) problem.

Formally describing stage I, let us define S_i^u as the set of unpacked signals at iteration i for a given ECU. This problem will be solved for each ECU and hence, the ECU superscript is dropped for clarity. At first iteration, $i = 0$ we have $S_0^u = \mathbf{S}$. At any iteration i, the procedure is as follows. For each type of CAN-FD frame $1 \leq k \leq m$, we compute the minimum bandwidth wastage that is possible with a subset of the unpacked signals S_i^u, by solving the CaFeS problem (note that the m CAN-FD frame types are due to the number of available frame sizes in CAN-FD). After solving the CaFeS problem for each type of CAN-FD frame $1 \leq k \leq m$, we select the CAN-FD frame that has the minimum bandwidth waste and remove the packed signals from S_i^u to obtain S_{i+1}^u. This process is iterated until all signals have been packed. When the iterations terminate, it gives us a set of bandwidth optimized and packed CAN-FD frames.

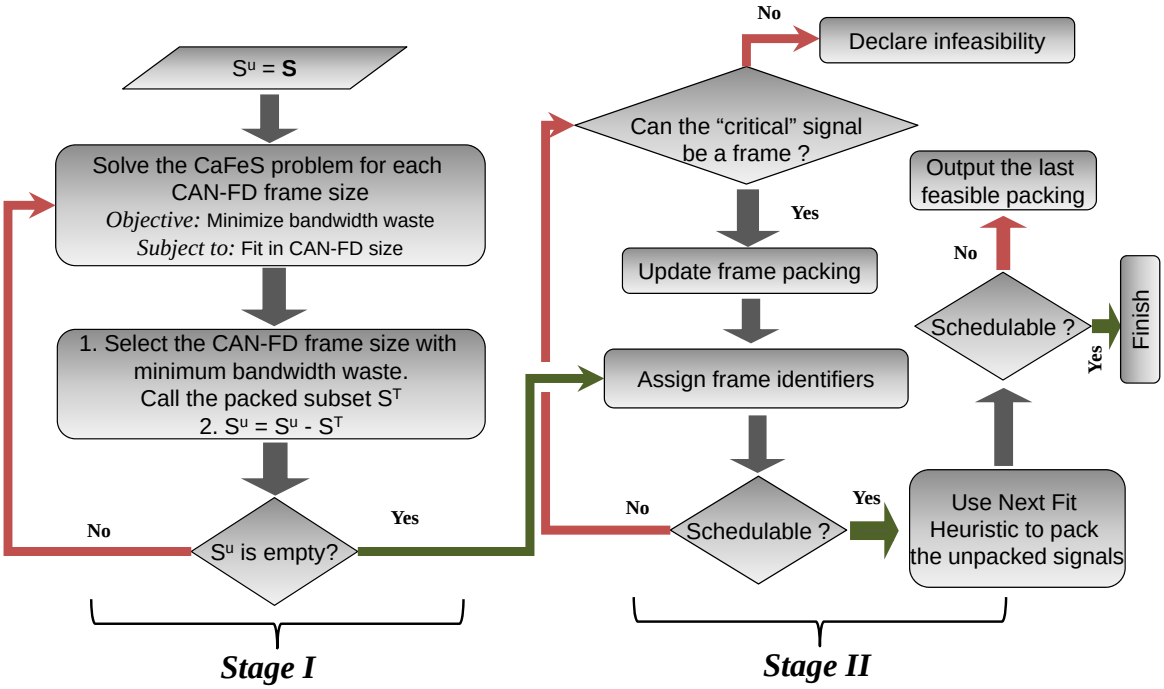

Fig. 3. Overview of our approach. Stage I searches the design space for a bandwidth optimized packing. Stage II assigns priority to the packed frames and verifies schedulability. If the optimized packing is unschedulable, stage II iterates to find a less optimized but schedulable solution. In this case, stage II also makes a final pass in an attempt to improve bandwidth utilization.

The objective function for bandwidth waste of the CaFeS problem should explicitly account for the portion of the CAN-FD payload size that has not been packed by any signal because we wish to minimize this portion in order to maximize the amount of bandwidth we spend on signal communication. It should also account for any bandwidth waste because of heterogeneous periods of the constituent signals.

The CaFeS problem: ***CAN-FD Frame Selection*** (CaFeS) is the core problem solved at each iteration for each CAN-FD frame type and is defined below. The subscript i for the current iteration is dropped here.

$$\text{Minimize} \quad \frac{\left(W_k - \sum_{j=1}^{|\mathbf{S}^u|} x_j l_j\right) + OH(W_k)}{H} \tag{8}$$

$$H = \min \bigcup_{j=1}^{|\mathbf{S}^u|} \{x_j h_j\} \setminus \{0\} \tag{9}$$

$$\text{Subject to} \sum_{j=1}^{|\mathbf{S}^u|} x_j l_j \leq W_k \tag{10}$$

Above, W_k, l_j and h_j, respectively, are the CAN-FD frame size of the kth frame, the length of the payload size of the jth signal and the period of the jth signal. The boolean variable x_j is set to one if the jth signal is selected for packing and is set to zero otherwise. Given this, it may be now seen that the optimization function (Equation 8) essentially captures the minimum bandwidth loss that one may achieve while packing a subset of the signals into a CAN-FD of type k. $W_k - \sum_{j=1}^{|\mathbf{S}|} l_j x_j$ is the bandwidth loss that occurs due to the fact that the selected signals might not completely fill the CAN-FD frame. The overhead $OH(W_k)$ is the overhead in bits that must be paid for every frame that is packed. OH is a function similar to Equation 2 and is used to compute the extra bits, apart from the payload, that are required to transmit the CAN-FD frame of size W_k. Including OH in the objective function is important because it formally captures the motivation from Figure 2 that overheads have a significant impact on the bandwidth utilization. The optimization objective also captures the bandwidth loss that might occur on account of varying periods of the constituent signals. Equation 9 highlights the fact that the period for the resulting frame will be minimum of all the periods of the selected signals. Equation 10 is the constraint that all the selected signals must fit into the CAN-FD frame under consideration.

Solving the CaFeS problem: We propose a dynamic programming algorithm that solves this optimization problem *optimally* and is listed in Algorithm 1. Line 2 calls a function called maxSize() that returns an upper bound UB on the total size if all signals are packed together, regardless of the CAN-FD frame. The next few lines initialize the variables to be computed by the recursive equations of the dynamic programming algorithm. $w_{x,y}$ is the minimum bandwidth waste, computed in same way as in Equation 8, but with the constraint that one considers the signals 1 to x out of the n signals and that the total sum of the sizes of the signals may be at most y. We also have two additional variables $p_{x,y}$ and $s_{x,y}$ that, respectively, denote the period of the frame and the summation of the sizes of the signals in the frame. $p_{x,y}$ and $s_{x,y}$ are considered valid if and only if $w_{x,y}$ has a finite value. It may be easily verified that Algorithm 1 runs in pseudo-polynomial time in $O(UB \times n^2)$.

Lines 6 to 28 of the algorithm are the two main loops of the algorithm. The outer loop corresponds to each new signal s_x that is being considered. The inner loop corresponds to a *maximum* possible frame size y from 0 to UB. Note that this frame size does not correspond to the CAN-FD frame size but rather only to the sum of the sizes of the constituent signals. At each iteration x, y, the values

Algorithm 1 CAN-FD Frame Selection

Input: The signal set $\mathbf{S}^u$ to be packed in the ith iteration and the capacity W_k of the CAN-FD frame currently under consideration.

1: $n \leftarrow |\mathbf{S}^u|$
2: $UB \leftarrow maxSize(\mathbf{S}^u, n)$
3: **for** $y \leftarrow 0$ to UB **do**
4: $w_{0,y} \leftarrow \infty$, $p_{0,y} \leftarrow \infty$, $s_{0,y} \leftarrow 0$
5: **end for**
6: **for** $x \leftarrow 1$ to n **do**
7: **for** $y \leftarrow 0$ to UB **do**
8: **if** $y - l_i < 0 || y > W_k$ **then**
9: $w \leftarrow \infty$, $p \leftarrow \infty$, $s \leftarrow 0$
10: **else**
11: $s = s_{x-1,y-l_{x-1}} + l_x$
12: $p = min\{p_{x-1,y-l_{x-1}}, H_x\}$
13: $w = \frac{(W_k - s) + OH}{H_x}$
14: **end if**
15: **if** $w < 0$ **then**
16: $w_{x,y} \leftarrow \infty$
17: **else**
18: $w_{x,y} \leftarrow min(w_{x-1,y}, w)$
19: **end if**
20: **if** $w < w_{x-1,y}$ **then**
21: $p_{x,y} = p$
22: $s_{x,y} = s$
23: **else**
24: $p_{x,y} = p_{x-1,y}$
25: $s_{x,y} = s_{x-1,y}$
26: **end if**
27: **end for**
28: **end for**
29: Return $\min\{w_{n,y} \,|\, y \leftarrow 1, 2, \ldots, UB\}$

$p_{x,y}$, $s_{x,y}$ and $w_{x,y}$ are updated. If the cell is infeasible, then line 9 marks them accordingly. Lines 11 to 13 compute the temporary values p, w and s assuming that the signal s_x is now packed. However, if this packing does not lead to better bandwidth when compared to bandwidth utilization $w_{x-1,y}$, obtained by the previously known packing with size y, we retain the previously computed value $w_{x-1,y}$ (lines 20 to 25). We conclude the algorithm description with the note that, essentially, the CaFeS dynamic programming (DP) algorithm builds a two dimensional table of size $(n+1) \times (UB+1)$. Each cell of the DP table contains three values corresponding to the signals packed. The contents for the (x,y)th cell are the bandwidth waste $(w_{x,y})$, the period $(p_{x,y})$ and size $(s_{x,y})$, henceforth referred to as the DP tuple.

CaFeS problem — An Example: We explain the intuition behind the working of the above DP algorithm with an example. The example is deliberately kept simple for ease of understanding and for illustrating a DP table that may fit into the spatial constraints in this paper. We consider two signals s_1 and s_2 that are to be packed with sizes $l_1 = 3$, $l_2 = 2$ and with periods $h_1 = 10$ and $h_2 = 10$. Here $n = 2$ and $UB = 2 \times \max(l_1, l_2) = 6$. Let us assume that the overhead OH is 1 for all frame sizes in this example. The DP table size in this case will be $(n+1) \times (UB+1) = 3 \times 7$.

Let us consider that the bin capacity of the CAN-FD frame currently considered is 5. Table I shows the DP table built for this example. For any $1 \leq x \leq 2$ and $1 \leq y \leq 6$, the (x,y)th cell of this table shows the values of $w_{x,y}$ on top followed by $s_{x,y}$ and $p_{x,y}$ (in this order) in the bottom. Recall that smaller the value of $w_{x,y}$, the better is the bandwidth usage of the CAN-FD frame. The row corresponding to $x = 0$ and the first column corresponding to $y = 0$ are maintained for initialization purposes that allows the recursive equations in the DP algorithm to work seamlessly. In all the cells in the first row and first column, the DP tuple is assigned the following values $w_{x,y} = \infty$, $s_{x,y} = 0$ and $p_{x,y} = \infty$ or, in short, $(\infty, \infty, 0)$ which means an infeasible solution (lines 8 to 9 in Algorithm 1). The value corresponding to all the cells in the column $y = 6$ are also assigned $(\infty, \infty, 0)$ because $(y = 6) > (W_k = 5)$, i.e., no frame size greater than the bin capacity is feasible.

In the second row (x = 1), we consider the first signal only. Here, the cells (1,1) and (1,2) are assigned infeasible values. This is because the first signal has size 3 and thus, there can be no frame that considers this signal and has size less than 3, thereby, ruling out the cells with $y = 1$ and $y = 2$. When $y = 3$ at cell $(1,3)$, it is feasible to consider a frame that consists of just the signal itself. Thus, the cell $(1,3)$ in the DP table has a valid tuple. The value 0.30 in the tuple comes from the fact that $(W_k - l_1 + OH)/p_1 = 3/10 = 0.3$. The other values in the tuple are equal to the period and the size of the signal. Similarly, the cells $(1,4)$ and $(1,5)$ allow a frame with only the first signal because $y = \{4,5\} > 3$. This is reflected in the value of the tuples.

In row 2 (x=2), we now consider both the signals. Again, the cell with $y = 1$ is infeasible because none of the signals can be part of a frame that has a size 1. However, $j = 2$ now becomes feasible because the second signal with size $l_2 = 2$ can be a frame of size 2 by itself. The tuple here reflects the values of such a frame. For example, the optimization objective has a value $(W_k - l_2 + OH)/p_2 = 4/10 = 0.2$. The cell $(2,3)$ is interesting because here we have two possibilities — a frame consisting only of s_1 (tuple corresponding to cell $(1,3)$) or a frame consisting only of s_2. The frame consisting of only s_1 leads to less waste and hence, the cell $(2,3)$ is updated to reflect this. The same is true for the cell $(2,4)$. At cell $(2,5)$, however, it now becomes feasible to consider both signals and the tuple in this cell must be updated to reflect this. Thus, this cell must contain the best packing considering that both signals may be packed. A frame consisting of both signals will have a bandwidth utilization of $(W_k - (l_1 + l_2) + OH)/min(p_1, p_2) = 1/10 = 0.1$. Compared to the value in $1,3$ (considering only s_1), this is more optimized and hence, the cell $(2,5)$ is updated to reflect this. Note that the DP algorithm returns the minimum from the last line of the DP algorithm (line 29). This ensures that we have considered all possibilities - s_1 as a frame, s_2 as a frame and both of them as a frame. In our example, the minimum value will be the tuple from cell $(2,5)$ that represents bandwidth waste of 0.1. This shows that in this example, with bin capacity of 5, it is best to pack these two signals together into one frame.

B. Stage Two: Schedulability

After completion of stage I, we have a set of packed frames. However, the frames have not been assigned priorities. In stage II of our heuristic, immediately following stage I (see Figure 3), the frames are assigned priorities such that the set of packed frames are schedulable. Towards this, we follow existing state-of-art technique [2], where the interested reader may find the details of the priority

$x\uparrow$	$y \rightarrow$						
	0	**1**	**2**	**3**	**4**	**5**	**6**
2	∞, ∞, 0	∞, ∞, 0	0.4, 2, 10	0.3, 2, 10	0.3, 2, 10	0.1, 2, 10	∞, ∞, 0
1	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	0.3, 3, 10	0.3, 3, 10	0.30, 3, 10	∞, ∞, 0
0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0	∞, ∞, 0

TABLE I
DP TABLE FOR TWO SIGNALS WITH SIZE 3 AND 2 AND WITH PERIODS OF 10.

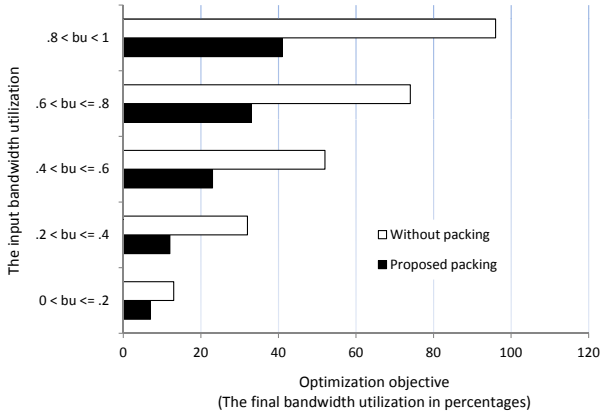

Fig. 4. Comparing the improvements delivered by our framework.

assignment technique as well as the schedulability analysis. However, the priority assignment may declare infeasibility meaning that no feasible assignment exists such that the given set of frames may be scheduled feasibly.

As shown in Figure 3, the next step is to identify the critical frame(s). For instance, a frame might be infeasible because of a signal that has a small deadline. Removing this signal from the frame will result in two frames. It relaxes the deadline of the original frame and leads to new signal with only one frame. This may not be optimized with regards to bandwidth utilization but might now lead to a feasible schedule. Our heuristic follows such a strategy in an iterative loop (Figure 3) until a feasible schedule is found or until all signals are unpacked, in which case it declares infeasibility.

If a feasible solution is reached after unpacking signals, our heuristic makes a final pass to pack together the newly unpacked signals. This packing is now based on a Next-Fit heuristic that greedily attempts to put signals with similar bandwidth utilization together into one frame. If such a packing fails, our heuristic reverts back to the previously known feasible solution.

VII. Experiments

A. Experimental Setup

All the experiments were conducted on a OS X (version 10.8.5) machine running on a 2.7GHz Intel Core i7 processor with 4GB main memory. The input benchmarks were generated by varying the signal parameters like periods and payload size with a uniform distribution in order to cover a wide range of possible scenarios. With uniform distribution, the periods and the signals were varied between $100ms$ and $5000ms$ and, respectively, between 1 byte and 14 bytes.

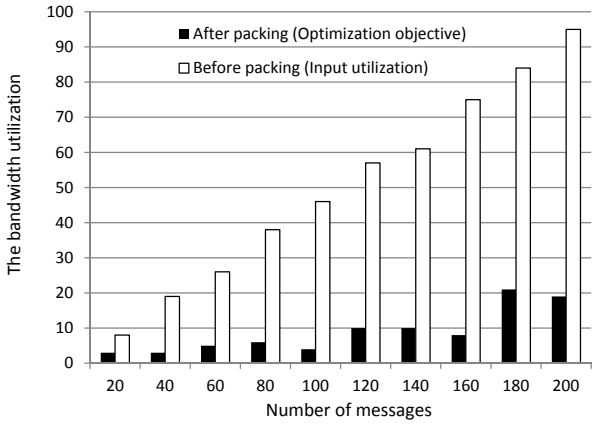

Fig. 5. The improvements obtained by our framework considering that all signals have same periods, i.e., bandwidth is saved only due to intelligent selection of CAN-FD frame sizes.

B. Benefits of Packing

In the first set of experiments, we proceed to show the benefits of packing the signals together in CAN-FD. We compare the bandwidth utilization delivered by our framework compared to the bandwidth utilization by unpacked signals. Note that when signals are not packed together, each signal is a frame in itself and the size of the resulting frame is the nearest CAN-FD frame size that may accommodate it. We varied the number of signals provided as an input between 20 and 200. For each input size, we generated 100 benchmarks with the parameters described above. We show the results in Figure 4.

To illustrate the results, we have categorized the input according their input bandwidth utilization. The input bandwidth utilization is the summation of ratios of the transmission times to the periods of the input signals. The y-axis highlights the 5 categories with the input bandwidth utilization varying between 0 - 20%, 20 - 40%, 40 - 60%, 60 - 80% and 80 - 100%. For each category, we plot two bars - one showing the average bandwidth utilization obtained by our framework while the other showing the average bandwidth utilization obtained without any packing, i.e., each signal is a frame. The benchmarks that were found to be unschedulable (less than 5% of the total benchmarks were unschedulable) by our framework were discarded and not included in the results shown. The results reinforces the fact that packing may lead to significant improvements over strategies that consider each signal as an independent frame - no matter what is the input bandwidth utilization.

It is noteworthy that the cases when the input utilization exceeds 60% to 70% have a resulting bandwidth of around 40% after packing. Reducing such high input utilizations without frame packing may require system redesign, possibly adding new CAN-FD subsystems. Such design practices may lead to unnecessary complexity, which can be avoided if intelligent frame packing—such as the one proposed in this paper—is used to optimize bus utilization.

C. Exploiting the CAN-FD Frame Size

One of the key insights of this paper is that the disparate CAN-FD frame sizes must be carefully chosen when signals are packed into frames. In order to evaluate the strength of our proposed heuristic to tackle this problem effectively, we conducted experiments where all signals in the generated

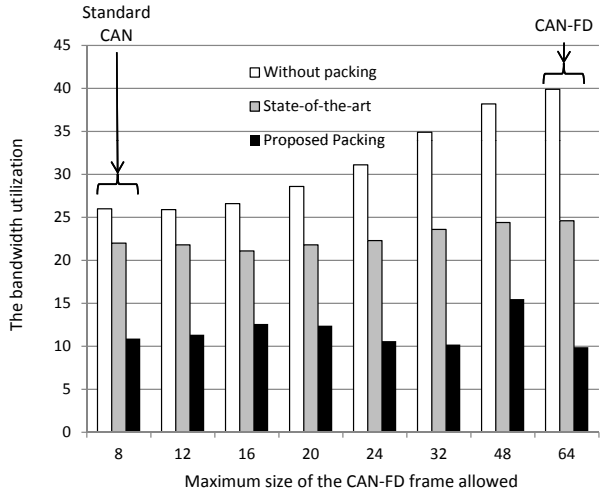

Fig. 6. The improvements obtained by our framework over the state-of-the-art [11] frame packing algorithm.

signals were set to the same period (which was randomly generated for each signal set). The only source of bandwidth waste will be partial packing of the available CAN-FD frame sizes. Thus, the results will test our heuristic with respect to its effectiveness in managing the variable CAN-FD frame sizes for bandwidth efficient packing.

Figure 5 shows the increase in input utilization as the number of signals increase (white columns). The black columns show the bandwidth utilization that is obtained after packing signals into frames by our heuristic. We can observe that our heuristic delivers highly significant bandwidth optimization even in this experiment; in fact, the relative improvement increases as the number of signals increase, which shows the importance of applying frame packing for communication-intensive systems.

D. Comparing with the State-of-the-art

Since there is no previous work addressing CAN-FD frame packing, we compare our approach to the best known results for standard CAN. Several techniques for frame packing have been proposed for standard CAN and the best known results were published in [11]. All heuristics for standard CAN assume 8 bytes as the largest frame size available (protocol restriction in standard CAN).

For all possible CAN-FD frame sizes from 8 till 64, we apply their algorithm and the results are shown in Figure 6. In these experiments, to be fair, we provide the frame sizes of (i.e, 8, 12, 16, 20, 24, 32, 48, and 64 bytes) as a bound to the previous heuristics [11] and restrict our heuristic to the same upper bound. The results compare the bandwidth utilization obtained when (i) there is no packing (each signal is a frame), (ii) the packing is obtained by the state-of-the-art [11] and (iii) the packing is obtained by our heuristic. It may be noted that the first point on the x-axis (8 bytes) refers exactly to the frame packing problem for standard CAN. As the graph shows, our heuristic delivers 51% improvement in bandwidth utilization over the best known heuristic for standard CAN. The rest of the results show that while the state-of-the-art can provide significant savings over unpacked signals if applied to CAN-FD, it is outperformed by our heuristic significantly (our heuristic delivers an average of around 45% improvement). The last point on the x-axis represents our unmodified heuristic for CAN-FD (without restrictions on payload size), with improvements reaching 59% with our proposed heuristic.

E. Scalability

The runtime complexity of our heuristic is dominated by the CaFeS problem, which is pseudo-polynomial. In practice, our tool completed the entire flow in less than couple of seconds (1857 milliseconds) for each benchmark. Considering the fact that these experiments were conducted on a laptop and without any specific optimization to utilize heterogeneous multi-core CPUs or GPUs, we conclude that scalability is not an issue for our framework.

VIII. Conclusion

This paper motivated the need for frame packing in CAN-FD protocol arising from the variable sized frame capacities, formulated the problem, and proposed an efficient solution that exploits the particular new features of CAN-FD. The results show that our frame packing strategy not only can be used to optimize CAN-FD bus utilization and increase future growth margin significantly, but also that frame packing is essential in cases where the system to be deployed is very communication intensive.

Acknowledgements

The authors thank Harald K. Eisele from Adam Opel AG for giving a great introduction to CAN-FD and for providing valuable feedback on the work presented in this paper.

References

[1] T. G. Crainic, G. Perboli, W. Rei, and R. Tadei. Efficient lower bounds and heuristics for the variable cost and size bin packing problem. *Comput. Oper. Res.*, 38(11), 2011.

[2] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien. Controller area network (CAN) schedulability analysis: Refuted, revisited and revised. *Real-Time Systems*, 35(3):239–272, 2007.

[3] M. Di Natale, H. Zeng, P. Giusto, and A. Ghosal. *Understanding and Using the Controller Area Network Communication Protocol.* Springer, 2012.

[4] H. Eisele and K.P. Orlando. What CAN-FD offers for automotive networking? In *Stuttgart International Symposium on Automotive and Engine Technology*, 2014.

[5] D K Friesen and M A Langston. Variable sized bin packing. *SIAM J. Comput.*, 15(1):222–230, 1986.

[6] M. Haouari and M. Serairi. Heuristics for the variable sized bin-packing problem. *Computers and Operations Research*, 36(10):2877 – 2884, 2009.

[7] F. Hartwich. CAN with flexible data-rate. In *International CAN Conference*, 2012.

[8] J. Kang and S. Park. Algorithms for the variable sized bin packing problem. *European Journal of Operational Research*, 147(2):365 – 372, 2003.

[9] F. D. Murgolo. An efficient approximation scheme for variable-sized bin packing. *SIAM J. Comput.*, 16(1):149–161, 1987.

[10] Bosch White Paper. CAN with flexible data rate. http://www.bosch-semiconductors.de/, 2011.

[11] F. Polzlbauer, I. Bate, and E. Brenner. Optimized frame packing for embedded systems. *Embedded Systems Letters*, 4(3):65–68, 2012.

[12] P. Pop, P. Eles, and Z. Peng. Schedulability-driven frame packing for multicluster distributed embedded systems. *ACM Trans. Embed. Comput. Syst.*, 4(1):112–140, 2005.

[13] R. Saket and N. Navet. Frame packing algorithms for automotive applications. *J. Embedded Computing*, 2(1), 2006.

[14] K. Sandstrom, C. Norstom, and M. Ahlmark. Frame packing in real-time communication. In *RTCSA*, 2000.

[15] B. Tanasa, U. D. Bordoloi, P. Eles, and Z. Peng. Reliability-aware frame packing for the static segment of Flexray. In *EMSOFT*, 2011.

[16] Vijay V. Vazirani. *Approximation Algorithms.* Springer, 2002.