The Multi-Domain Frame Packing Problem for CAN-FD

Prachi Joshi[1], Haibo Zeng[2], Unmesh D. Bordoloi[3], Soheil Samii[4], S. S. Ravi[*5], and Sandeep K. Shukla[6]

1 **Virginia Tech, Blacksburg, VA, USA**
prachi@vt.edu
2 **Virginia Tech, Blacksburg, VA, USA**
hbzeng@vt.edu
3 **General Motors, USA**
unmesh.bordoloi@gm.com
4 **General Motors, USA; and Linköping University, Linköping, Sweden**
soheil.samii@gm.com
5 **Virginia Tech,Blacksburg, VA, USA; and University at Albany – SUNY, NY, USA**
ssravi@vt.edu
6 **IIT Kanpur, Kanpur, India**
sandeeps@cse.iitk.ac.in

Abstract

The Controller Area Network with Flexible Data-Rate (CAN-FD) is a new communication protocol to meet the bandwidth requirements for the constantly growing volume of data exchanged in modern vehicles. The problem of frame packing for CAN-FD, as studied in the literature, assumes a single sub-system where one CAN-FD bus serves as the communication medium among several Electronic Control Units (ECUs). Modern automotive electronic systems, on the other hand, consist of several sub-systems, each facilitating a certain functional domain such as powertrain, chassis and suspension. A substantial fraction of all signals is exchanged across sub-systems. In this work, we study the frame packing problem for CAN-FD with multiple sub-systems, and propose a two-stage optimization framework. In the first stage, we pack the signals into frames with the objective of minimizing the bandwidth utilization. In the second stage, we extend Audsley's algorithm to assign priorities/identifiers to the frames. In case the resulting solution is not schedulable, our framework provides a potential repacking method. We propose two solution approaches: (a) an Integer Linear Programming (ILP) formulation that provides an optimal solution but is computationally expensive for industrial-size problems; and (b) a greedy heuristic that scales well and provides solutions that are comparable to optimal solutions. Experimental results show the efficiency of our optimization framework in achieving feasible solutions with low bandwidth utilization. The results also show a significant improvement over the case when there is no cross-domain consideration (as in prior work).

1998 ACM Subject Classification C.3 [Special-Purpose and Applicaton-Based Systems] Real-Time and Embedded Systems

Keywords and phrases frame packing, CAN-FD, integer linear programming, Audsley's algorithm

Digital Object Identifier 10.4230/LIPIcs.ECRTS.2017.12

[*] S. S.Ravi was supported in part by NSF Grants DIBBS ACI-1443054 and BIG DATA IIS-1633028.

1 Introduction

Modern automotive electronic systems consist of several sub-systems, each facilitating a certain functional domain such as powertrain, chassis, suspension, steering, etc. Over the years, there has been a steady increase in the number of messages exchanged among such *sub-systems*, also called *domains* in the paper. A few reasons for this trend are: 1) increase in the number of features enabled by software and electronics; 2) integration of functionalities onto System-on-Chip (SoCs) allowing up-integration of hardware capacity into fewer (but more computationally capable) Electronic Control Units (ECUs); and 3) consolidation of ECUs for cost reduction. The proliferation of such cross-domain traffic can significantly contribute to bandwidth bottlenecks, and as we show in this paper, it leads to a non-trivial optimization problem. In this paper, we propose a frame packing algorithm for multiple sub-systems that are served by CAN-FD (Controller Area Network with Flexible Data-Rate) field buses. To the best of our knowledge, this is the first attempt to formulate and propose a solution to what we term as the problem of *multi-domain frame packing for CAN-FD*.

Since its development in the 1990s, CAN (Controller Area Network) has attracted a significant amount of research from the real-time systems community. The CAN protocol adopts a collision detection and resolution scheme, where the message to be transmitted is chosen according to its identifier. When multiple nodes need to transmit over the same bus, the message with the lowest identifier is selected for transmission. This arbitration protocol allows encoding of the message priority into the identifier field and the implementation of priority-based scheduling. The analysis of the CAN message response time [22, 5] was derived using an analogy to the results on CPU scheduling, providing an exact evaluation and a safe approximation of the worst-case message response times.

In recent years, automotive features have been growing, thereby demanding an increase in bandwidth requirements of the communication network. In order to bridge the gap between CAN and other higher data rate communication protocols (such as TTEthernet, MOST150, etc.), two major improvements were added to CAN to develop CAN-FD [9]: 1) the increase of bit-rate (up to 8 Mbps); and 2) the increase of payload sizes (up to 64 bytes). The physical layer of CAN was unchanged: it still uses a bitwise arbitration method of contention resolution based on message identifiers.

Related Work. As mentioned earlier, the frame packing problem has been considered in the literature only for a single domain in CAN-FD. Bordoloi and Samii [2] present a dynamic programming approach for packing the signals followed by a priority assignment step. Urul's thesis [23] points out that schedulability of frames can be improved by packing same period signals in each frame. Di Natale et al. [16] present a single-step Integer Linear Programming (ILP) formulation to achieve both optimal bandwidth utilization and schedulability. However, its applicability is limited to medium-size problems. In [25], the authors map signals to frames on CAN as part of their task allocation and priority assignment problem to optimize end-to-end latency using an MILP.

The frame packing problem is related to the classical bin packing problem (BPP) which is known to be NP-hard. For CAN-FD, a particularly relevant subclass of BPP is the *variable-sized bin packing problem* (VSBPP). While the classical bin packing problem has been studied extensively (e.g., [8]), VSBPP has received relatively less attention. Friesen and Langston [7] propose and formally analyze the performance of three heuristics for VSBPP. Murgolo [15] presents a polynomial-time approximation scheme for VSBPP. We note that the approximation algorithms for VSBPP cannot be directly applied to the frame packing problem for CAN-FD since the goal of the latter problem is to minimize bandwidth utilization instead of the number of bins (frames). In addition, the frame packing problem must consider both the size and the period of each signal.

For standard CAN, both [17] and [20] present frame packing approaches inspired by the *next fit decreasing* heuristic for BPP. The difference is that the algorithm in [17] sorts the signals according to their periods, while the one in [20] sorts them based on their deadlines. Saket and Navet [19] present a frame packing heuristic which sorts the signals by their bandwidth utilization and then packs this list of sorted signals alternately from both sides of the list (to increase the chances of signals with similar periods to be packed together).

The frame packing problem has also been considered under other communication protocols that are time-triggered (such as FlexRay static segment [13, 24, 21, 11, 3]) or mixed event/time-triggered [18]. The nature of these communication protocols, and thus the frame packing problem, is very different from that for CAN and CAN-FD. Hence, the corresponding approaches and results are not directly applicable here.

Contributions. The frame packing problem in CAN and CAN-FD has been addressed in the literature [2, 23, 16]. However, unlike our paper, these references consider only a single domain. Even for a single domain, the problem is already challenging: the above references have pointed out its relationship to the bin packing problem which is known to be NP-hard.

In this paper, we study the multi-domain frame packing problem for CAN-FD, and develop a two-stage optimization procedure. In the first stage, we propose an ILP based approach to generate an optimal solution for frame packing as well as a heuristic that scales to large problem sizes. Our ILP and heuristic approaches capture the details of inter-domain communication and gating over multiple CAN-FD networks. In the second stage, we propose an extension to Audsley's algorithm [1] for optimal priority assignment with multi-domain frames. In case the priority assignment does not lead to a feasible solution, we provide an effective strategy to re-pack the frames. We conduct experiments on synthetic systems (whose characteristics are close to real systems) and show that our heuristic runs extremely fast compared to the computationally expensive (in terms of both time and memory) ILP and yet returns solutions that are on average within 3% of those produced by the ILP in terms of bandwidth utilization per domain. Our experiments also show that the repacking strategy is effective in that it often leads to schedulable solutions. Compared to the approach without cross-domain consideration, our approach can typically save 6%–10% bandwidth utilization per domain.

The rest of the paper is organized as follows. Section 2 provides a brief overview of the CAN-FD protocol. Section 3 defines the multi-domain frame packing problem. Section 4 provides an overview for the two-stage iterative framework. Section 5 presents the approach using ILP formulation and Section 6 describes the greedy heuristic algorithm. Section 7 provides the experimental results and compares the ILP and heuristic approaches. Finally, Section 8 summarizes our contributions and presents some concluding remarks.

2 CAN-FD Overview

In this section, we briefly describe the main features of CAN-FD. The CAN-FD frame format is shown in Figure 1. For a more detailed description, readers are referred to [2]. Like CAN, a **dominant** bit is a logical 0 and a **recessive** bit is a logical 1. As in the figure, a CAN-FD frame is partitioned into two phases: arbitration phase and data phase.

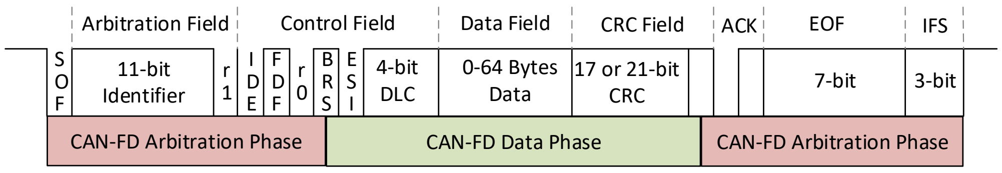


Figure 1 CAN-FD Frame Format (from [9]).

Arbitration Phase. The arbitration phase in the CAN-FD frame contains the following fields: SOF (Start Of Frame), arbitration, part of the control field, ACK (Acknowledgment), EOF (End OF Frame), and IFS (Inter-Frame Space). The 11-bit (or 29-bit in case of extended format) identifier represents the priority of the frame: the lower the value of the identifier, the higher the priority. The arbitration for transmission happens as follows. During the idle state of the bus, all the nodes with some ready frames send the 11-bit identifier after the SOF bit. During the transmission of the identifier bits, if a node transmits a recessive bit but finds a dominant bit on the bus, it stops transmission due to the presence of a higher priority node contesting for transmission. In the end, the node with the highest priority message wins the arbitration and continues the transmission.

The transmission of bits in the arbitration phase occurs at the arbitration bit-rate, and the duration of transmission for each bit is denoted as t_a. For example, if the arbitration rate is chosen as 500 Kbps, then $t_a = 2\mu s$.

Data Phase. The BRS (Bit-Rate Switch) bit is one of the additions to the CAN-FD frame format. It is used to decide whether the bit-rate in the data phase is the same as that of the arbitration phase (BRS = 0) or it switches to the increased bit rate (BRS = 1). Since our focus is on CAN-FD, we consider the BRS bit in the frames to be recessive (i.e., BRS = 1). At the increased rate of data transmission, each bit transmission occurs with a duration denoted by t_d. For example, if the data rate is chosen as 2 Mbps, $t_d = 0.5\mu s$. The 4-bit DLC (data-length code) field specifies the payload size (in bytes) of the data field. CAN-FD offers 16 distinct payload sizes: 0 through 8, 12, 16, 20, 24, 32, 48 and 64 bytes.

The data field is followed by the Cyclic Redundancy Check (CRC) field, which has 17 bits for payloads up to 16 bytes, and 21 bits otherwise. The CRC delimiter bit (recessive) is transmitted next. After this, the bit rate is changed back to that of the arbitration phase.

Transmission Time. The worst-case transmission time (WCTT) of a CAN-FD frame is a function of its payload size (i.e., the size of the data field) and the data rates. As in [2], if p is the payload size (in bytes) of a CAN-FD frame, its WCTT is given by:

$$\mathrm{WCTT}(p) \;=\; 32\,t_a + \left(28 + 5\left\lceil \frac{p-16}{64} \right\rceil + 10p\right) t_d\,. \tag{1}$$

In this work we have assumed the arbitration and data rates to be same for all the domains, however our approach can be easily adapted to the scenario where each domain/network has a different bit-rate. The bit-rates affect the WCTT expression (Equation 1), and therefore for the latter case we can compute the WCTT for each domain and use it in the bandwidth calculation corresponding to the domain. Similarly the schedulability analysis can be updated with the inclusion of the appropriate WCTT expression for determining the response time for a frame on each domain.

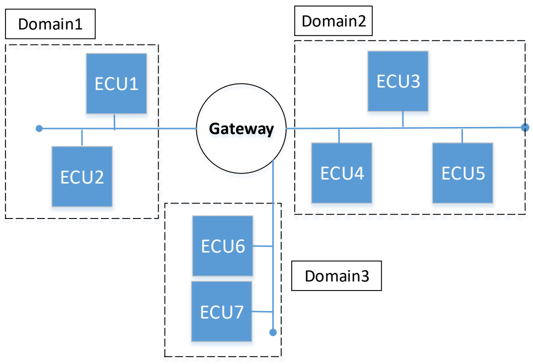


Figure 2 A multi-domain CAN-FD architecture.

Table 1 Parameters of each signal and frame.

Notation	Significance
$t(\sigma)$	Period of signal σ
$d(\sigma)$	Deadline of signal σ
$p(\sigma)$	Size (in bytes) of signal σ
$\delta(\sigma)$	Domains of signal σ

Notation	Significance
$T(\gamma)$	Period of frame γ
$D(\gamma)$	Deadline of frame γ
$P(\gamma)$	Payload size (in bytes) of frame γ
$\Delta(\gamma)$	Set of domains of signals packed in frame γ
$\mathcal{S}(\gamma)$	Set of signals packed in frame γ
$C(\gamma)$	Worst-case transmission time (WCTT) of frame γ
$\pi(\gamma)$	Priority level of frame γ

3 Problem Definition

We assume a network topology where several CAN-FD sub-systems, typically serving different domains, are connected to a central gateway. We use the terms "domain" and "sub-system" interchangeably. The ECUs in each domain generate signals which must be packed into frames and transmitted to their destination domains. Each domain uses a CAN-FD bus for data communication. The gateway is responsible for forwarding the frames to their respective destination domains *without repacking or reassigning frame identifiers*. Such an architecture is relevant in the automotive industry [10]. Figure 2 provides an example of a system where 3 domains are connected by a gateway.

Table 1 summarizes the parameters of each signal and frame. In the following, we define the problem and review the schedulability analysis for CAN-FD.

Problem Description: Let $\mathbb{D} = \{\Delta_1, \Delta_2, \ldots, \Delta_{|\mathbb{D}|}\}$ denote the set of *domains*. Let n_i denote the number of ECUs in domain Δ_i, $1 \leq i \leq |\mathbb{D}|$. The jth ECU from domain Δ_i is denoted by $\psi_{i,j}$. The set of signals generated by ECU $\psi_{i,j}$ is represented by $\mathcal{S}(\psi_{i,j}) = \{\sigma_{i,j}^k \mid k = 1, \ldots, |\mathcal{S}(\psi_{i,j})|\}$. Each signal $\sigma \in \mathcal{S}(\psi_{i,j})$ is specified as a quadruple $\langle\, t(\sigma), d(\sigma), p(\sigma), \delta(\sigma)\,\rangle$, whose components denote respectively the period, deadline, size (in bytes) and domains (including the source and destinations) of signal σ.

The output of the multi-domain frame packing problem is a set of frames $\Gamma = \{\gamma_1, \gamma_2, \ldots\}$ satisfying *all* of the following conditions.

1. Each signal σ is placed in exactly one frame γ.
2. For each frame γ, all the signals in γ are from the *same* ECU, and the periods of all the signals in γ are **harmonic**[1] (i.e., $\forall \sigma_i, \sigma_j \in \gamma$, there exists an integer k such that either $t(\sigma_i) = k \cdot t(\sigma_j)$ or $t(\sigma_j) = k \cdot t(\sigma_i)$).
3. The sum of the sizes of all signals in a frame γ is at most the payload size of γ.
4. The packed frames are schedulable in all the domains in which they are transmitted.

Each frame γ is characterized by a tuple $\langle \mathcal{S}(\gamma), \Delta(\gamma), T(\gamma), D(\gamma), P(\gamma), C(\gamma), \pi(\gamma) \rangle$, where $\mathcal{S}(\gamma)$ is the set of signals packed into γ and $\Delta(\gamma)$ is the set of domains of the signals in γ. The quantities $T(\gamma)$, $D(\gamma)$, $P(\gamma)$, and $C(\gamma)$ are respectively the period, deadline, payload size (in bytes), and WCTT of the frame γ. Given $\mathcal{S}(\gamma)$, the other parameters of the frame γ are determined as follows.

- $\Delta(\gamma) = \bigcup_{\sigma \in \mathcal{S}(\gamma)} \delta(\sigma)$; i.e., the set of domains of γ is the union of those for all the signals in γ.
- $T(\gamma) = \gcd\{t(\sigma) \,:\, \sigma \in \mathcal{S}(\gamma)\}$; i.e., the period of γ is the greatest common divisor (gcd) of the periods of the signals in γ.
- $D(\gamma) = \min\{d(\sigma) \,:\, \sigma \in \mathcal{S}(\gamma)\}$; i.e., the deadline of γ is the smallest deadline among the signals in γ.
- $P(\gamma) \geq \sum_{\sigma \in \mathcal{S}(\gamma)} p(\sigma)$; i.e., the payload of γ is large enough to contain its constituent signals. The CAN-FD standard [9] restricts $P(\gamma)$ to be one of the following values: 0 through 8, 12, 16, 20, 24, 32, 48 and 64 bytes.
- The WCTT $C(\gamma)$ of a frame γ is determined by Equation (1), with the variable p being replaced by $P(\gamma)$.
- $\pi(\gamma)$ represents the unique priority (across all domains) assigned to frame γ.

The bandwidth utilization $U(\gamma)$ of frame γ is defined as

$$U(\gamma) \;=\; \frac{C(\gamma)}{T(\gamma)}\,. \tag{2}$$

The objective is to minimize the total bandwidth utilization over all the domains. This is motivated by extensibility to accommodate possible future functions [2].

CAN-FD Schedulability: The schedulability analysis for CAN-FD follows that of CAN [5], where the worst-case response time is always inside the busy period. The busy period of priority level-i is a contiguous interval of time that starts at the critical instant, during which any frame of priority lower than γ_i is unable to win arbitration. The length of the busy period $L(\gamma_i)$ and the index $q^{\max}(\gamma_i)$ of the last instance are calculated as

$$L(\gamma_i) = B(\gamma_i) + \sum_{j \in hp(i) \bigcup \{i\}} \left\lceil \frac{L(\gamma_i)}{T(\gamma_j)} \right\rceil C(\gamma_j), \qquad q^{\max}(\gamma_i) = \left\lceil \frac{L(\gamma_i)}{T(\gamma_i)} \right\rceil \tag{3}$$

[1] Note that our approach is valid even without this condition.

where $hp(i)$ is the set of frames with priority higher than γ_i, and $B(\gamma_i)$ is the *blocking time*, i.e., the maximum time spent on waiting for the transmission of a lower priority message already on the bus when γ_i becomes ready.

The response time $R(\gamma_{i,q})$ of the q-th instance $\gamma_{i,q}$ in the busy period is given by

$$R(\gamma_{i,q}) = w(\gamma_{i,q}) - (q-1)T(\gamma_i) + C(\gamma_i) \tag{4}$$

where q ranges from 1 to the last instance $q^{\max}(\gamma_i)$ of γ_i inside the busy period. The worst-case queuing delay $w(\gamma_{i,q})$ for the q-th instance in the busy period is

$$w(\gamma_{i,q}) = B(\gamma_i) + (q-1)C(\gamma_i) + \sum_{j \in hp(i)} \left\lceil \frac{w(\gamma_{i,q})}{T(\gamma_j)} \right\rceil C(\gamma_j)\,. \tag{5}$$

In Equation (5), $w(\gamma_{i,q})$ appears on both sides. However, the right hand side is a monotonic non-decreasing function of $w(\gamma_{i,q})$. Hence, $w(\gamma_{i,q})$ can be solved using the iterative procedure defined by the equation below.

$$w^{n+1}(\gamma_{i,q}) = B(\gamma_i) + (q-1)C(\gamma_i) + \sum_{j \in hp(i)} \left\lceil \frac{w^n(\gamma_{i,q})}{T(\gamma_j)} \right\rceil C(\gamma_j)\,. \tag{6}$$

The calculation can start with an initial value of $w^0(\gamma_{i,q}) = B(\gamma_i) + (q-1)C_i$, and stop when $w^{n+1}(\gamma_{i,q}) = w^n(\gamma_{i,q})$ or $w^{n+1}(\gamma_{i,q}) - (q-1)T(\gamma_i) + C(\gamma_i) > D(\gamma_i)$, the latter condition indicating that γ_i is unschedulable. The worst-case response time of γ_i, denoted by $R(\gamma_i)$, is the maximum among all its instances in the busy period; that is,

$$R(\gamma_i) = \max_{q=1,\ldots,q^{\max}(\gamma_i)} \{R(\gamma_{i,q})\} \tag{7}$$

4 Overview of the Two-Stage Optimization Procedure

The multi-domain frame packing problem for CAN-FD is NP-hard. This follows directly from the NP-hardness of the special case of single-domain frame packing problem [2]. To cope with this complexity, we consider an optimization procedure that consists of two stages, as illustrated in Figure 3. In the first stage, we try to find a signal-to-frame packing with minimum total bandwidth utilization. In the second stage, given the signal-to-frame packing, we perform priority assignment to all the frames such that the response time of each frame falls within its deadline. We choose to minimize the total bandwidth in the first stage for two reasons: one is that this is also the overall objective, the other is that smaller bus bandwidth utilization generally leads to better schedulability. However, the latter is not always the case. When a candidate frame packing produced by Stage 1 is not schedulable, all the frames or a subset thereof are repacked, this time focusing on improving schedulability. The two-stage procedure iterates until a feasible solution is found or after a certain number of iterations of repacking have been completed.

As discussed in subsequent sections, we present two instantiations of the optimization procedure. One instantiation formulates the problem in the first stage as an integer linear program (ILP) and leverages existing solvers to find an optimal solution (i.e., one with minimum total bandwidth utilization). In case of unschedulability, the second stage iteratively produces another packing by sacrificing optimality by a certain amount, with additional constraints in the ILP model. As will be seen in Section 5, the ILP formulation is somewhat intricate due to the discontinuous nature of the frame sizes available under CAN-FD. Due to the number of constraints in the ILP formulation, this approach does not scale to large

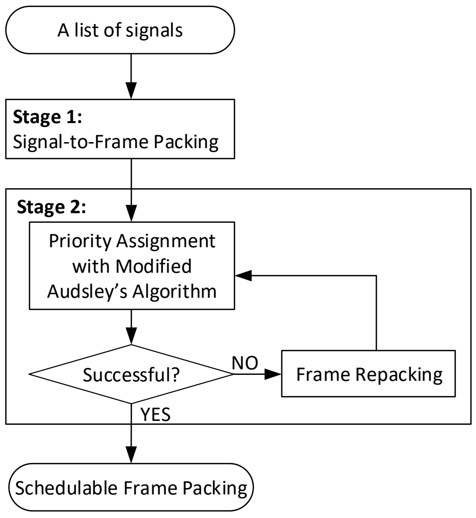


Figure 3 The two-stage optimization procedure.

systems. The second instantiation of the procedure in Figure 3 uses a fast heuristic in the first stage. Based on the observation that the first stage, namely the problem of signal-to-frame packing, is related to the bin packing problem, we develop a bandwidth-based best-fit (greedy) approach, where each signal is packed into a candidate frame that minimizes the total bandwidth utilization over all the domains. The second stage directly repacks a selected list of frames in case of unschedulability.

The problem of frame priority assignment can be solved efficiently. We propose a modified version of the Audsley's algorithm [1] that only needs to check a quadratic number of candidate priority assignments. Similar to Audsley's algorithm, it iteratively picks a frame that can be assigned a particular priority level starting from the lowest priority. However, when choosing a candidate frame, it should guarantee that assigning the priority does not violate the schedulability in any domain to which the frame will be transmitted. Here, the priority order of frames remains the same across all domains, as the gateway is assumed to forward the frames without repacking or reassigning frame identifiers.

5 ILP-based Approach

5.1 ILP formulation for Signal-to-Frame Packing

Since each frame may only contain signals sent by the same ECU and we do not consider schedulability in the first stage, it is sufficient to perform frame packing for each ECU separately. Hence, we present the ILP formulation considering the set of signals $\mathcal{S}(\psi_{i,j})$ generated by ECU $\psi_{i,j}$. The total bandwidth utilization is the sum of the utilization values over all the ECUs.

We generate a set of virtual frames $\Gamma_{i,j} = \{\gamma_l \mid l = 1, ..., |\mathcal{S}(\psi_{i,j})|\}$, one for each signal in $\mathcal{S}(\psi_{i,j})$. Thus, $\Gamma_{i,j}$ consists of the maximum number of frames that could be used for packing the signals in $\mathcal{S}(\psi_{i,j})$; a packing may use only a subset of $\Gamma_{i,j}$. Each virtual frame $\gamma_l \in \Gamma_{i,j}$ is represented as a tuple $\langle T(\gamma_l), D(\gamma_l), P(\gamma_l)\rangle$, which specifies respectively the period, deadline and size of the frame. Here, we fix the period of each frame to be the period of its corresponding signal. Thus, $\forall l$, $T(\gamma_l) = t(\sigma_l)$. However, the deadline $D(\gamma_l)$ and size $P(\gamma_l)$

depend on the signals packed in γ_l. We use a binary *parameter* to denote whether a signal shall be transmitted in a particular domain.

$$Y_{k,e} = \begin{cases} 1 & \text{if the destination of } \sigma_k \text{ is in domain } \Delta_e \\ 0 & \text{otherwise.} \end{cases}$$

Note that this information is available as part of the input. Thus, $Y_{k,e}$ is *not* a variable in the ILP formulation. We define a binary (decision) variable $x_{k,l}$ to indicate the mapping of signals to frames:

$$x_{k,l} = \begin{cases} 1 & \text{if signal } \sigma_k \text{ is packed into frame } \gamma_l \\ 0 & \text{otherwise.} \end{cases}$$

Each signal should be assigned to one and only one frame:

$$\forall k: \sum_{l=1}^{|\mathcal{S}(\psi_{i,j})|} x_{k,l} = 1 \tag{8}$$

The period of a frame should be a divisor of the period of any signal assigned to the frame:

$$\forall k, l \text{ such that } t(\sigma_k) \bmod T(\gamma_l) \neq 0: \quad x_{k,l} = 0 \tag{9}$$

If two signals have non-harmonic periods, they should not be packed in the same frame:

$$\forall k, m \text{ such that } t(\sigma_k) \geq t(\sigma_m) \bigwedge t(\sigma_k) \bmod t(\sigma_m) \neq 0,\ \forall \gamma_l: \ x_{k,l} + x_{m,l} \leq 1 \tag{10}$$

Since some frames may not be assigned any signals during the packing, we must ensure that they are not taken into account while computing the bandwidth utilization. To do this, we use a binary variable ρ_l to indicate if γ_l contains any signals. Hence,

$$\frac{\sum_{k=1}^{|\mathcal{S}(\psi_{i,j})|} x_{k,l}}{|\mathcal{S}(\psi_{i,j})|} \leq \rho_l \bigwedge \rho_l \leq \sum_{k=1}^{|\mathcal{S}(\psi_{i,j})|} x_{k,l} \tag{11}$$

For each frame γ_l, we introduce a variable z_l that represents the sum of the sizes (in bytes) of the signals packed in γ_l:

$$z_l = \sum_{k=1}^{|\mathcal{S}(\psi_{i,j})|} x_{k,l} \cdot p(\sigma_k) \tag{12}$$

The maximum size of a frame is 64 bytes. Hence, the total size of the signals assigned to a frame should be no more than 64 bytes:

$$\forall l: \ z_l \leq 64 \tag{13}$$

As described in Section 2, CAN-FD allows 16 different frame payload sizes (0 through 8, 12, 16, 20, 24, 32, 48 and 64 bytes). Hence, the size $P(\gamma_l)$ of any frame γ_l can be modeled as a discontinuous function defined by

$$P(\gamma_l) = \begin{cases} z_l, & 0 \leq z_l \leq 8 \\ 12, & 8 < z_l \leq 12 \\ 16, & 12 < z_l \leq 16 \\ 20, & 16 < z_l \leq 20 \\ 24, & 20 < z_l \leq 24 \\ 32, & 24 < z_l \leq 32 \\ 48, & 32 < z_l \leq 48 \\ 64, & 48 < z_l \leq 64 \end{cases}$$

We define eight new binary variables λ_1, λ_2, …, λ_8, which determine the ranges of the z_l variables.

$$\begin{cases} z_l \le 8 + M(1-\lambda_1) & \\ z_l \le 12 + M(1-\lambda_2) & \bigwedge\ z_l + M(1-\lambda_2) > 8 \\ z_l \le 16 + M(1-\lambda_3) & \bigwedge\ z_l + M(1-\lambda_3) > 12 \\ z_l \le 20 + M(1-\lambda_4) & \bigwedge\ z_l + M(1-\lambda_4) > 16 \\ z_l \le 24 + M(1-\lambda_5) & \bigwedge\ z_l + M(1-\lambda_5) > 20 \\ z_l \le 32 + M(1-\lambda_6) & \bigwedge\ z_l + M(1-\lambda_6) > 24 \\ z_l \le 48 + M(1-\lambda_7) & \bigwedge\ z_l + M(1-\lambda_7) > 32 \\ z_l \le 64 + M(1-\lambda_8) & \bigwedge\ z_l + M(1-\lambda_8) > 48 \end{cases} \tag{14}$$

where M is a large enough constant. Hence, the size of a frame can be expressed as

$$P(\gamma_l) = \lambda_1 \cdot z_l + 12\lambda_2 + 16\lambda_3 + 20\lambda_4 + 24\lambda_5 + 32\lambda_6 + 48\lambda_7 + 64\lambda_8\,. \tag{15}$$

However, there is a product term, namely $\lambda_1 \cdot z_l$, in Equation (15). This can be linearized by introducing a new variable v_l as follows.

$$v_l = \lambda_1 \cdot z_l \ \ \Rightarrow\ \ v_l \ \le\ z_l + M(1-\lambda_1) \ \bigwedge\ z_l \ \le\ v_l + M(1-\lambda_1) \ \bigwedge\ v_l \ \le\ M \cdot z_l\,. \tag{16}$$

Therefore, Equation (15) can be rewritten as the following linear constraint:

$$P(\gamma_l) = v_l + 12\lambda_2 + 16\lambda_3 + 20\lambda_4 + 24\lambda_5 + 32\lambda_6 + 48\lambda_7 + 64\lambda_8\,. \tag{17}$$

To calculate the WCTT of frame γ_l using Equation (1), we note that the ceiling function $\left\lceil \frac{P(\gamma_l)-16}{64} \right\rceil$ can only take on two values: 0 if $P(\gamma_l) \le 16$ and 1 otherwise. Hence, we introduce a binary variable u_l to represent it. The constraints on u_l are as follows:

$$P(\gamma_l) + M(1-u_l) \ >\ 16 \ \bigwedge\ P(\gamma_l) \ \le\ 16 + M \cdot u_l\,. \tag{18}$$

Now, the expression for WCTT becomes

$$C(\gamma_l) \ =\ 32\,t_a + (28 + 5u_l + 10P(\gamma_l))\,t_d\,. \tag{19}$$

The total bandwidth utilization for all the frames for an ECU $\psi_{i,j}$ in the CAN-FD network can be expressed as follows:

$$\sum_{l}^{|\mathcal{S}(\psi_{i,j})|} \left[\rho_l \cdot \frac{C(\gamma_l)}{T(\gamma_l)} + \sum_{e \ne i} \left(\eta_{l,e} \cdot \frac{C(\gamma_l)}{T(\gamma_l)}\right)\right] \tag{20}$$

where the first part $\rho_l \cdot \frac{C(\gamma_l)}{T(\gamma_l)}$ corresponds to the bandwidth utilization over the source domain Δ_i, and the second part $\sum_{e \ne i}\left(\eta_{l,e} \cdot \frac{C(\gamma_l)}{T(\gamma_l)}\right)$ corresponds to the bandwidth utilization over all the destination domains. In Equation (20), $\eta_{l,e}$ is a binary variable to determine whether the frame γ_l has any signal with a destination in domain Δ_e. Using the binary parameter $Y_{k,e}$ defined earlier, $\eta_{l,e}$ can be defined as

$$\eta_{l,e} = \begin{cases} 1 & \text{if } \sum_{k=1}^{|\mathcal{S}(\psi_{i,j})|}(x_{k,l} \cdot Y_{k,e}) \ge 1 \\ 0 & \text{otherwise} \end{cases}$$

The linear constraints that enforce the definition of $\eta_{l,e}$ are as follows:

$$\frac{\sum_{k=1}^{|\mathcal{S}(\psi_{i,j})|}(x_{k,l} \cdot Y_{k,e})}{|\mathcal{S}(\psi_{i,j})|} \ \le\ \eta_{l,e} \ \le\ 1 \ \bigwedge\ \eta_{l,e} \ \le \sum_{k=1}^{|\mathcal{S}(\psi_{i,j})|} (x_{k,l} \cdot Y_{k,e})\,. \tag{21}$$

Algorithm 1 Modified Audsley's Algorithm for Multi-Domain CAN-FD

```
1: procedure AudsleyMultiDomain ($\Gamma$)
2:     Let $N = |\Gamma|$, Create a list $Q$ containing all the frames in $\Gamma$
3:     for $\pi = N$ downto 1 do
4:         for each frame $\gamma \in Q$ do
5:             flag_found = FALSE
6:             for each $\Delta_i \in \Delta(\gamma)$ do
7:                 $R(\gamma)$ = ComputeResponseTime ($\gamma$, $\Delta_i$, $\Gamma$)
8:             if $\gamma$ is schedulable in all $\Delta_i \in \Delta(\gamma)$ then
9:                 Assign priority level $\pi$ to $\gamma$
10:                Remove $\gamma$ from $Q$
11:                flag_found = TRUE
12:                break
13:        if flag_found is FALSE then
14:            Report unschedulability and return
15:    Report schedulability
```

The objective is to minimize the total bandwidth utilization of all the frames:

$$\min \sum_{i}^{|\mathbb{D}|} \sum_{j}^{n_i} \sum_{l}^{|\mathcal{S}(\psi_{i,j})|} \left[\rho_l \cdot \frac{C(\gamma_l)}{T(\gamma_l)} + \sum_{e \neq i} \left(\eta_{l,e} \cdot \frac{C(\gamma_l)}{T(\gamma_l)} \right) \right]. \tag{22}$$

In Equation (22), $\rho_l \cdot \frac{C(\gamma_l)}{T(\gamma_l)}$ and $\eta_{l,e} \cdot \frac{C(\gamma_l)}{T(\gamma_l)}$ are both a product of a binary variable and a real variable. They can be linearized in a manner similar to that of Equation (16).

5.2 Modified Audsley's Algorithm for Priority Assignment

In order to assign priority identifiers to all the frames, we extend Audsley's algorithm [1] to the multi-domain case (Algorithm 1). The input to the algorithm is the set of frames Γ for all the domains. Similar to Audsley's algorithm, priority levels are assigned iteratively to all the frames starting from lowest to highest (Lines 3–15). At each iteration, a priority level is assigned to the first frame γ that satisfies the schedulability constraints over all the domains belonging to $\Delta(\gamma)$ (Lines 8–12). If a priority level cannot be assigned to any of the frames (i.e., flag_found is FALSE), the algorithm reports unschedulability (Lines 13–14). If all the frames are assigned a unique priority, the algorithm is successful in finding a schedulable priority assignment.

Using the approach in [6], it can be easily shown that the schedulability of a multi-domain frame (i.e., whether it meets the deadline requirement in all its domains) satisfies all the three conditions which are necessary and sufficient to provide an optimal priority assignment. Hence, our extension to Audsley's algorithm for the multi-domain CAN-FD system is optimal for finding a schedulable priority assignment.

5.3 Handling Infeasibility

Although in principle our frame packing scheme supports schedulability (since it minimizes bandwidth utilization which indirectly helps to reduce network traffic), there are cases when the modified Audsley's algorithm returns infeasibility.

In the case of ILP, when we encounter infeasibility, we call the solver again after relaxing the optimal value of the objective function (by doubling the optimality gap in each iteration) and setting a time limit of one hour for each iteration. We report infeasibility if no feasible priority assignment is found even after a given number of iterations.

6 Greedy Algorithm-based Approach

The ILP approach discussed in the previous section provides an optimal packing of the signals into frames with respect to bandwidth utilization. However, due to its exponential time complexity, it does not scale well to large sets of signals. Therefore, we propose a greedy heuristic (Algorithm 2) for the frame packing step. The heart of the algorithm presents the steps for packing the signals from one ECU; the outermost loop ensures that the steps are iterated over all the ECUs.

6.1 Description of the Heuristic for Signal-to-Frame Packing

The algorithm first sorts the input signals for each ECU (Line 3 in Algorithm 2) on the basis of a parameter such as the period, size or the input bandwidth utilization (which is given by the size/period) of signals. It then uses a Bandwidth Best-Fit approach to pack the signals into frames as follows. Starting from the first signal, each signal is placed in a frame that minimizes the total bandwidth utilization of the system (*over all the domains*). The steps shown in lines 6 and 7 create a new frame and add the signal to it. To obtain the bandwidth utilization for a frame, we use Equation (2) to compute the utilization over each of its destination domains and then take their sum. The total bandwidth utilization of the system is the sum of the bandwidth utilization over all the frames (Equation (20)). Further, lines 8–14 compute and store the total bandwidth utilization by temporarily adding the signal to an existing frame. Before a signal is assigned to an existing frame, the "if(σ_k can be added to F_j)" condition (Line 9 in Algorithm 2) checks (i) whether the frame can accommodate the new signal (i.e., the total size of all the signals in the frame is at most 64); and (ii) whether the period of the new signal is harmonic with the periods of the other signals in that frame. The steps in lines 15 and 16 decide whether it is beneficial to add the current signal to a new frame or to one of the existing frames. The output of the algorithm is a list of frames (Γ) which stores the frames created in each step.

The quality of the packing depends on the sorting criterion used in Line 3. In our experiments, we compare different sorting methods using each of the above parameters (i.e., period, size and size/period) in both increasing and decreasing orders.

Time Complexity Analysis: For each ECU ψ_i, we show that the above heuristic runs in $O(s^2 f)$ time, where $s = |S_i|$ is the number of signals for the ECU and f is the number of frames in the resulting packing. To begin with, sorting the set $\mathcal{S}(\psi_i)$ can be done in $O(s \log s)$ time. Now, for each signal $\sigma \in \mathcal{S}(\psi_i)$, the time for finding the best placement into a frame can be estimated as follows. Let $\Gamma(\psi_i) = \{\gamma_1, \gamma_2, \ldots, \gamma_r\}$ denote the current set of frames when σ is considered. Creating a new frame containing just σ can be done in $O(1)$ time. As mentioned above, testing whether σ can be added to a frame γ_i involves two checks involving the size of the frame and the harmonicity of periods of the signals currently in the frame. It is easy to see that each of these checks can be done in time $O(|\gamma_i|)$, where $|\gamma_i|$ denotes the number of signals in γ_i. Thus, the total time for checking whether σ can be added to each of the existing frames is $O(\sum_{i=1}^{r} |\gamma_i|) = O(s)$, since all the frames together contain at most s signals. For each placement of σ in a frame, it is also easy to see that computing the

Algorithm 2 Greedy Algorithm

```
1: procedure GREEDY-BW-BEST-FIT ($\Psi$, $S$)
2:     for each ECU $\psi_i \in \Psi$ do
3:         Sort($\mathcal{S}(\psi_i)$)
4:         Number of frames $n = 0$, list of frames $\Gamma = \emptyset$
5:         for each signal $\sigma_k$ in $\mathcal{S}(\psi_i)$ do
6:             Create a new frame $F_{n+1}$ containing only $\sigma_k$
7:             Compute the total BW utilization $u_{n+1}$ of frames $F_1, ..., F_n, F_{n+1}$
8:             for $j = 1$ to $n$ do
9:                 if ($\sigma_k$ can be added to $F_j$) then
10:                    Add $\sigma_k$ to $F_j$
11:                    Compute the total BW utilization $u_j$ of frames $F_1, ..., F_n$
12:                    Remove $\sigma_k$ from $F_j$
13:                else
14:                    Set $u_j$ to infinity
15:            Find the smallest $u_j$ among $u_1, ..., u_{n+1}$ and pack $\sigma_k$ in $F_j$
16:            if ($j == n + 1$) then add $F_{n+1}$ to $\Gamma$ and set $n = n + 1$
17:    Return $\Gamma$
```

bandwidth utilization (as explained in the description of the heuristic) for signal σ can be done in $O(r + \sum_{i=1}^{r} |\gamma_i|) = O(s)$ time since $r \leq f \leq s$ and as observed earlier, $\sum_{i=1}^{r} |\gamma_i| \leq s$. As we need to compute the bandwidth utilization for at most $f + 1$ alternatives (including the new frame containing only σ), the time used for this step is $O(sf)$. In other words, for each signal, the greedy heuristic uses $O(sf)$ time. So, over all the s signals in $\mathcal{S}(\psi_i)$, the time complexity of the heuristic is $O(s^2 f)$.

6.2 Handling Infeasibility

In the second stage of the optimization, in case Algorithm 1 (i.e., the modified Audsley's Algorithm) fails to find a schedulable priority assignment, we propose a repacking method, with three variations, so that the frames may become schedulable. Our repacking strategy consists of two parts: the first part unpacks a selected set of frames based on certain conditions, and the second part repacks the signals removed from frames. The first part (unpacking of frames) is based on the following two methods.

1. **Unpacking Based on Destination Domains:** In case of a multi-domain system, Algorithm 1 reports infeasibility when a particular priority level cannot be assigned to any frame. This infeasibility could occur due to certain domains. Therefore, our first unpacking method attempts to separate the signals destined for different domains. The intuition behind such an unpacking can be understood from the following example. Consider a frame with 14 signals: σ_1, σ_2, ... σ_{14}. Suppose the first 12 signals have a total size of 40 bytes and their destination domain is D_1 while signals σ_{13} and σ_{14} have a total size of 2 bytes and their destination domain is D_2. Thus, such a frame carries an extra payload of 40 bytes to domain D_2. It could be beneficial to remove the signals intended for domain D_2 from the frame so that the overall schedulability (in particular for D_2) may be improved.
2. **Unpacking Based on Deadline:** Increasing the deadline of a frame is another method we adopt in order to satisfy the schedulability constraints. We apply this in our unpacking

scheme by separating the signals with the smallest deadline in a frame, thereby increasing the frame's deadline and its schedulability.

We apply at most one unpacking scheme per frame in an iteration. If the first criterion (destination domain based unpacking) is applicable to a frame, then we unpack the corresponding signals and do no further unpacking for this frame. If the first criterion does not apply to a frame (i.e., all the signals in the frame have the same destination domain), then we check the second criterion (deadline based unpacking). If neither of the criteria is met for a frame, we do not unpack that frame. After unpacking the signals, we repack them into existing frames using first-fit, best-fit and worst-fit heuristics. The repacking should satisfy the previously stated constraints of the problem (i.e., a frame should only have signals from the same ECU, all the signals in a frame should have harmonic periods and the total size of the signals in a frame should not exceed 64 bytes). The repacking step may suitably expand or shrink the size of a frame to handle the addition and removal of signals respectively.

We consider three different sets of frames as candidates for the unpacking and repacking steps. For each signal in the set, we unpack signals based on the above criteria and then repack them into existing frames (or generate new frames) using first-fit (FF), best-fit (BF) and worst-fit (WF) methods.

1. *All the frames*: In this case, we consider all the frames. We refer to this variation as the "All" heuristic.
2. *Unassigned frames from Audsley's method:* Here, we unpack only those frames for which Algorithm 1 could not assign a priority level. That is, at the priority level where Algorithm 1 fails to find a schedulable frame from the remaining set of frames, we consider this set for unpacking. We call this variation the "Unassigned" heuristic.
3. *Irreducible subset:* The idea behind computing an irreducible subset is similar to the computation of a minimal unsatisfiable core of a Boolean formula in conjunctive normal form [14]. When Algorithm 1 reports infeasibility, we compute a set of frames Γ', called an **irreducible subset**, satisfying the following condition: the set Γ' does not satisfy the schedulability constraints, but for any frame $\gamma \in \Gamma'$, the set $\Gamma' - \{\gamma\}$ becomes schedulable. We only unpack the frames which form an irreducible subset, based on the above two criteria. We refer this variation as "Irreducible subset" heuristic.

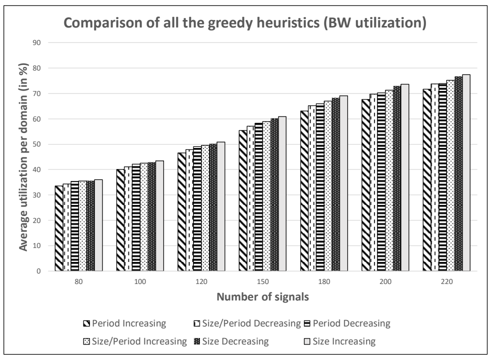
The reason for developing three variations of the repacking strategy is due to the complexity of making a given set of frames schedulable over the network. Since schedulability depends on a number of factors such as deadlines, traffic congestion over the network, sizes of frames, etc., there is no single factor which can be manipulated in order to obtain schedulability. The three different methods of repacking target different input sets. For example, for a particular input, it might be beneficial to unpack and repack a smaller set of "problematic" frames whereas another input might require a larger set of frames to be repacked. In the former case, the "Irreducible subset" approach could be more beneficial, and in latter case, the "All' approach might yield better results.

7 Experimental Results

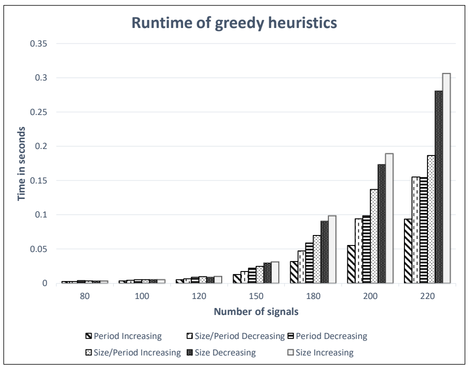
In this section, we present a detailed evaluation of the proposed algorithms using synthetic systems. For these experiments, we generated synthetic systems according to the guidelines on real-world automotive benchmarks [12], with minor modifications. Specifically, we redistributed the share of signals with size larger than 64 bytes to the bin "33-64 bytes" (as for this work we only consider signals with size up to 64 bytes), and the share of signals sent

Table 2 Signal parameters and their distribution.

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



(a) Comparison of the bandwidth utilization of all the proposed greedy heuristics.



(b) Comparison of the runtime for all the proposed greedy heuristics.

Figure 4 Comparison of bandwidth utilization and runtime of greedy algorithm.

by engine control tasks to those with periods between 1 and 20ms (as we do not consider the signals with angle-synchronous periods). Table 2 summarizes the distribution of signal periods and payload sizes used for generating the synthetic systems. Each signal is randomly assigned a source and a destination domain (from the set of domain IDs) with a probability of $1/|\mathbb{D}|$, where $|\mathbb{D}|$ is the total number of domains. Therefore, the probability of a signal being cross-domain (i.e., the probability that its source and destination domains are different) is $1 - 1/|\mathbb{D}|$. In all our experiments, we use a system with three domains, 10 ECUs (in total) and vary the number of signals from 80 to 220. Hence for our experiments, the probability of a signal being cross-domain is 2/3.

Our experiments are conducted using a high performance computing cluster at Virginia Tech. This cluster has 4 x E7-8867v4 2.4 GHz (Broadwell) processors and 3TB, 2400MHz memory on a Unix platform. We used IBM's CPLEX as the ILP solver and implemented the algorithms in C++.

7.1 Comparison of Greedy Packing Heuristics

For all of our experiments in this section, we generated 5000 benchmarks for each system size (in terms of the number of signals).

We first evaluate the different greedy packing heuristic approaches by comparing the bandwidth utilization they provide after packing. We note that our greedy approach

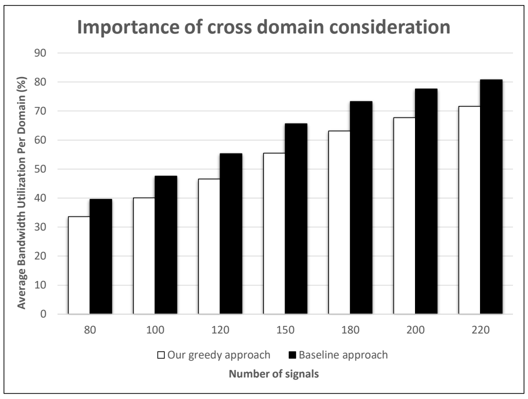


Figure 5 Comparing the greedy heuristic with a baseline packing approach which does not consider cross domain bandwidth while packing.

(Algorithm 2) leads to different bandwidth utilization values depending on the sorting criterion. We used the following parameters: period, size, and size/period (with increasing and decreasing orders). After the sorting step, each algorithm packs the signals in a greedy manner to optimize the bandwidth utilization.

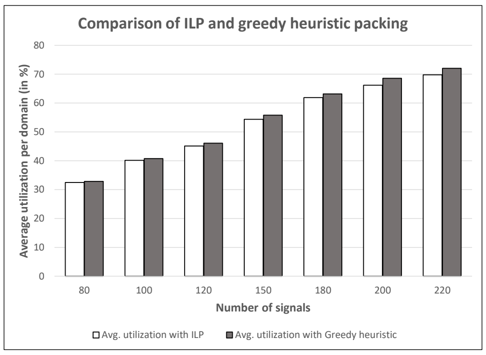
Figure 4a shows the bandwidth utilization per domain of the greedy heuristics, where the utilization is averaged over all those systems which are schedulable. As seen from the figure, there is small but noticeable variation in bandwidth utilization among the different heuristics. However, in all cases, sorting by increasing period performs the best. This is consistent with the observation that it is beneficial to pack signals with similar periods together, which in general reduces the total bandwidth utilization.

We also plot the runtime of the heuristics for each system size in Figure 4b. It is clear from this figure that the heuristic of sorting the signals in increasing order of periods has the smallest runtime; that is, it has an advantage over the other sorting approaches in terms of algorithmic efficiency as well. This is due to the fact that (as pointed out in the time complexity analysis for the greedy approaches) the runtime of the greedy algorithm is a function of the number of frames created, and the algorithm that sorts the signals in increasing order of periods creates the least number of frames for each signal size (as compared to the other heuristics). Hence, for the rest of the experiments, we use the heuristic that sorts the signals in increasing order of periods to represent all the greedy heuristics and compare it with the ILP-based approach.

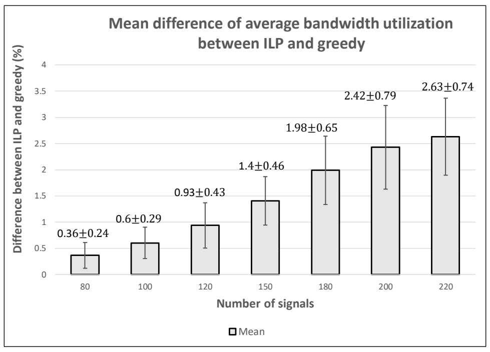
7.2 Importance of Cross-Domain Consideration

In this experiment, we demonstrate the importance of considering the cross-domain utilization during packing of the signals. Since there is no existing work for multi-domain frame packing in CAN-FD, we compare our greedy heuristic to a baseline approach which does not consider cross domain bandwidth utilization. Specifically, the baseline approach is the same as the greedy heuristic, except that it takes into account only the first part of Equation (20) (the source domain bandwidth utilization) but not the second part (cross-domain bandwidth utilization). For each system size, we tried 5000 benchmarks, and the bandwidth utilization represented is the average per domain over the 5000 benchmarks.

Figure 5 shows the significance of taking into account the cross-domain utilization for packing. We observe that there is a considerable reduction in bandwidth utilization per



(a) Greedy vs. ILP: bandwidth utilization per domain.



(b) Average and standard deviation for differences on utilization between ILP and greedy.

Figure 6 Comparison of ILP and greedy algorithms.

domain when frames are packed using our approach as opposed to the baseline approach: the typical reduction is in the range of 6% to 10% for utilization per domain. Also, the gap becomes larger as the number of signals increases.

7.3 Comparison of the Greedy Heuristic with ILP

In this experiment, we compare the ILP-based approach and the greedy heuristic in terms of their bandwidth utilization and the runtime. For these experiments, we used 100 synthetic systems for each size due to the excessively long runtime of ILP. We stopped at systems with 220 signals as ILP cannot scale to any larger systems: for systems with 4 domains, 15 ECUs and 250 signals, each of them takes about 7 hours on average.

Figure 6a illustrates the average bandwidth utilization per domain over the systems that are schedulable (either in the first packing attempt or after iteration/repacking). As can be observed from the figure, the bandwidth utilization of the greedy approach is quite close to that of ILP. The maximum mean difference on bandwidth utilization per domain is about 2.7% for systems with 220 signals.

Since Figure 6a gives only the average bandwidth utilization over all the systems, to better compare the ILP and the greedy approaches, we present the variability of the difference between the utilization values reported by them in Figure 6b. The gray bars represent the mean difference in percentage (over the 100 random systems, which are schedulable) between the bandwidth utilization (per domain) given by the ILP and the greedy approaches. The error bars represent the standard deviation of the difference. Figure 7 presents the distribution of the number of systems (that are schedulable) into different bins which represent the range of the difference between the average bandwidth utilization of the ILP and greedy approaches (as indicated in the legend). As the number of signals increases, the number of systems assigned to the larger bins also increases. Thus, Figures 6b and 7 show that with increasing number of signals, the difference in the bandwidth utilization given by the ILP and greedy approaches increases.

Figure 8a presents the average runtime of the systems for the ILP and greedy heuristic (in log scale). It is evident that the ILP would have scalability issues for larger systems, as the average runtime for the systems with 220 signals is already over 2.5 hours. The greedy approach on the other hand runs about 6 orders of magnitude faster than the ILP. Thus, we

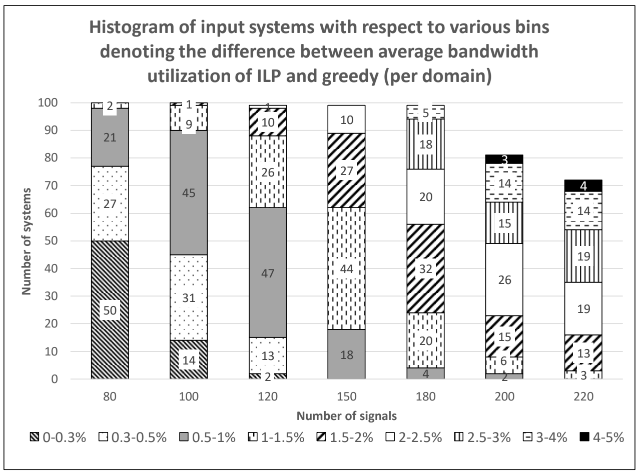
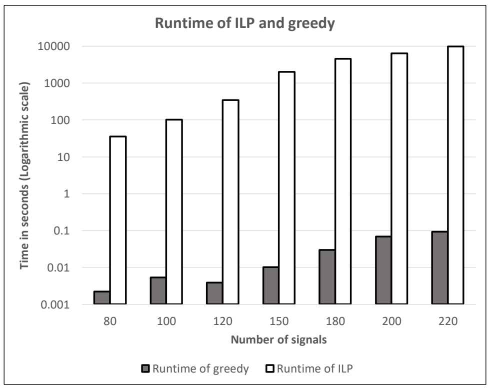
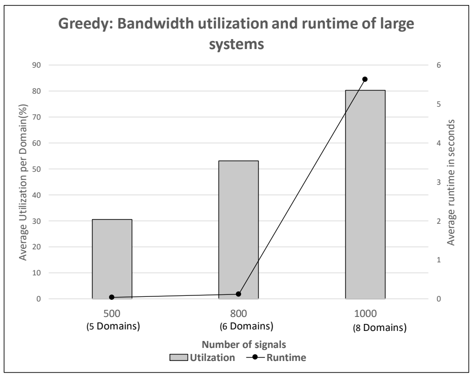


Figure 7 Distribution of systems within the various bin sizes (difference between greedy and ILP utilizations per domain).



(a) Greedy vs. ILP: runtime.



(b) Scalability of greedy: bandwidth utilization and runtime per domain for large sized systems.

Figure 8 Scalability of greedy with respect to ILP and industry sized systems.

can conclude that the greedy algorithm provides a packing whose bandwidth utilization is comparable to that of the ILP with a much smaller runtime. We note that in Figure 8a the average runtime of the greedy heuristic for 100 signals is slightly higher than that for the subsequent case of 120 signals. This is because one of the systems (out of 100) turned out to be unschedulable and thus the heuristic runs 10 iterations for this case, thereby increasing the average runtime. On the other hand, the unschedulable system in the case of 120 signals becomes schedulable in just one iteration with our heuristic (please refer to Figure 10).

In order to check the scalability of the greedy heuristic for industry sized systems we also conducted experiments (with 5000 systems) having 500, 800 and 1000 signals with 5, 6 and 8 domains and 5, 7 and 10 ECUs per domain respectively. We plot the bandwidth utilization and runtime in Fig.8b. The runtime of these systems was observed to be less than 6 seconds per system on the cluster (even in the largest size of 1000 signals), which shows that the

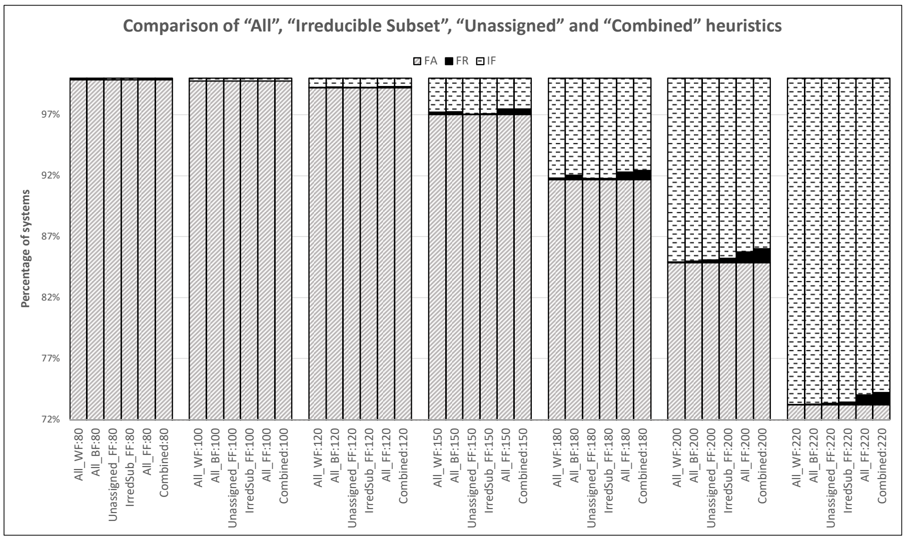


Figure 9 Comparison of "All","Irreducible subset","Unassigned" and "Combined" heuristics with respect to schedulability.

heuristic is easily scalable to large size systems. However for these experiments we used a slightly modified data generation scheme, where the contribution of signals from periods $1, 2$ and 5 ms which was 10% was distributed to signals with period 10 and 20 ms equally.

7.4 Handling Infeasibility

In this set of experiments, we compare the approaches for handling infeasibility for the greedy heuristic. We generated 5000 random systems for each system size (number of signals in the system). As described in Section 6.2, we have implemented three variations of the repacking heuristic, namely "All", "Unassigned" and "Irreducible subset", and each of these variations uses three types of repacking algorithms: first-fit (FF), best-fit (BF) and worst-fit (WF). We also combined all the three variations (with first-fit), which resulted in (slightly) better results with respect to feasibility. We refer to this heuristic as the "Combined" heuristic.

In the experiments, we find that FF consistently outperforms the other two repacking algorithms BF and WF. Among the three variations, "All" heuristic gives the best results most of the time with respect to the number of feasible systems after the iterative procedure. To minimize clutter, in Figure 9 we only present the results for six approaches: "All" with FF, WF and BF, "Unassigned" with FF, "Irreducible Subset" with FF, and "Combined" with FF. The rectangular bars represent the percentage of the total number of systems, the gray section gives the number of systems feasible in first attempt (FA), the black section gives the number of systems feasible after repacking (FR), and the section with horizontal dash pattern gives the number of infeasible systems (IF).

We observe that the scheme "All" with first-fit (labeled as All-FF) is able to get the maximum number of systems to become feasible after the iterative step (for each system size). This is due to the fact that the "Irreducible subset" and "Unassigned" heuristics unpack a subset of the frames unpacked by the "All" heuristic. Therefore "All" is able to remove potentially "problematic" signals from all the frames and thus provide more schedulable cases. Also, by combining all our repacking heuristics, more than 92% of the systems become

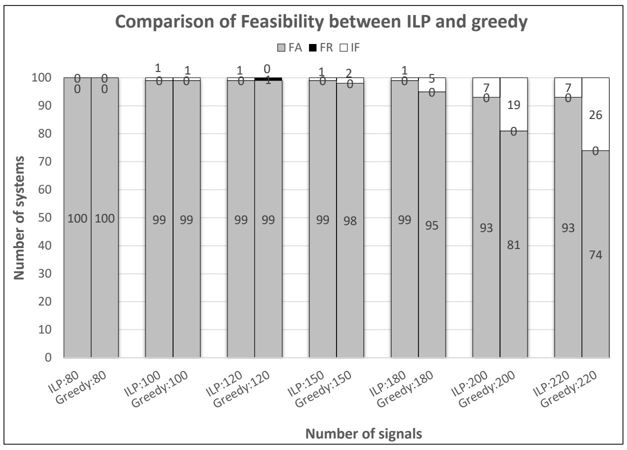


Figure 10 Infeasibility handling of ILP and greedy heuristic.

feasible (after repacking) for signal sizes 180 and below. For larger signal sizes, namely 200 and 220, 86% and 74% of the systems become feasible (after repacking) respectively.

Finally, we compare the infeasibility handling of the ILP and the greedy heuristic. For ILP, we use the iterative procedure described in Section 5.3, and for the greedy heuristic we use the "All" with first-fit scheme, since it was found to give the best results for infeasibility handling. Due to the long runtime of ILP, we generated 100 random systems for each system size. As in Figure 10, the performance of the greedy heuristic is comparable to that of ILP with respect to the number of feasible cases for small systems (namely, with number of signals below 150). However, for system size of 180 the greedy packing results in about 5% infeasible cases whereas the ILP delivers just 1% infeasible cases. Furthermore, for system size 200 and 220, the ILP gives 7% infeasible cases whereas the greedy approach gives 19% and 26% infeasible cases respectively. Due to its optimal packing (lower bandwidth utilization over the network), the ILP provides better feasibility at the cost of a much longer runtime.

8 Conclusions

In this paper, we motivate and propose solutions for the problem of frame packing for multi-domain CAN-FD systems. Existing work on frame packing for CAN-FD systems has not considered the problem from a multi-domain perspective. Our experiments show the significance of considering the multi-domain aspect (i.e., the inter-domain communication) for packing signals into frames. We present two approaches for the problem, namely ILP and greedy heuristic, both of which pack signals into frames with the goal of minimizing the bandwidth utilization over all the domains. In addition, we proposed an extension to Audsley's algorithm for assigning priority identifiers to the frames in the multi-domain case. In case of infeasibility, we developed several repacking heuristics so that the system may become feasible. Our experimental results show that the performance of the greedy heuristic is comparable to that of the ILP with respect to the bandwidth utilization as well as feasibility of the packed frames. However it is much faster than the ILP.

One line of future work is to investigate the frame packing problem for a heterogeneous multi-domain system where domains may be served by different communication protocols such as CAN, switched Ethernet, etc. In this work we have targeted optimization of bandwidth utilization which is an important metric for conserving network bandwidth (for future feature additions) and obtaining better schedulability. In our future work we would be interested in considering other aspects such as optimization of extensibility [26] and robustness [4].

Acknowledgments. The authors would like to thank Sudhakaran M. from GM R&D as the need for multi-domain problem formulation was triggered after an initial discussion with him on the opportunities for bandwidth optimization in automotive bus topologies. The work in this paper is supported by GM R&D. The authors also acknowledge Advanced Research Computing at Virginia Tech for providing computational resources and technical support that have contributed to the results reported within this paper.

References

1 N. C. Audsley. On priority assignment in fixed priority scheduling. *Information Processing Letters*, 79(1):39–44, 2001. `doi:10.1016/S0020-0190(00)00165-4`.

2 Unmesh Dutta Bordoloi and Soheil Samii. The frame packing problem for CAN-FD. In *Proceedings of the IEEE 35th IEEE Real-Time Systems Symposium (RTSS)*, pages 284–293, December 2014. `doi:10.1109/RTSS.2014.8`.

3 Armaghan Darbandi, Sungoh Kwon, and Myung Kyun Kim. Scheduling of time triggered messages in static segment of FlexRay. *International Journal of Software Engineering and its Applications*, 8(6):195–208, 2014. `doi:10.14257/ijseia.2014.8.6.16`.

4 Robert I. Davis and Alan Burns. Robust priority assignment for messages on controller area network (can). *Real-Time Systems*, 41(2):152–180, 2009. `doi:10.1007/s11241-008-9065-2`.

5 Robert I. Davis, Alan Burns, Reinder J. Bril, and Johan J. Lukkien. Controller area network (CAN) schedulability analysis: Refuted, revisited and revised. *Real-Time Systems*, 35(3):239–272, 2007. `doi:10.1007/s11241-007-9012-7`.

6 Robert I Davis, Liliana Cucu-Grosjean, Marko Bertogna, and Alan Burns. A review of priority assignment in real-time systems. *Journal of Systems Architecture*, 65:64–82, 2016. `doi:10.1016/j.sysarc.2016.04.002`.

7 Donald K. Friesen and Michael A. Langston. Variable sized bin packing. *SIAM Journal on Computing*, 15(1):222–230, 1986. `doi:10.1137/0215016`.

8 Michael R Garey and David S Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman and Company, San Francisco, CA, 1979.

9 Florian Hartwich. CAN with flexible data-rate. In *Proc. 13th International CAN Conference (iCC)*, pages 14:1–14:9. CAN in Automation (CiA), 2012.

10 Hogenmüller and Triess. Cost efficient gateway architecture for deterministic automotive networks, 2013. URL: `https://standards.ieee.org/events/automotive/12_Hogenmueller_Triess_EDE_Handout.pdf`.

11 Minkoo Kang, Kiejin Park, and Myong-Kee Jeong. Frame packing for minimizing the bandwidth consumption of the FlexRay static segment. *IEEE Transactions on Industrial Electronics*, 60(9):4001–4008, 2013. `doi:10.1109/TIE.2012.2208433`.

12 S. Kramer, D. Ziegenbein, and A. Hamann. Real world automotive benchmark for free. In *Workshop on Analysis Tools and Methodologies for Embedded and Real-Time Systems (WATERS)*, 2015.

13 Martin Lukasiewycz, Michael Glaß, Jürgen Teich, and Paul Milbredt. FlexRay schedule optimization of the static segment. In *Proceedings of the 7th IEEE/ACM international conference on Hardware/software codesign and system synthesis (CODES)*, pages 363–372. IEEE-ACM, 2009. `doi:10.1145/1629435.1629485`.

14 Inês Lynce and Joao P Marques-Silva. On computing minimum unsatisfiable cores. In *Proceedings of the International Symposium on Theory and Applications of Satisfiability Testing*, pages 305–310, 2004.

15 Frank D Murgolo. An efficient approximation scheme for variable-sized bin packing. *SIAM Journal on Computing*, 16(1):149–161, 1987. doi:10.1137/0216012.

16 Marco Di Natale, Celso Luiz Mendes da Silva, and Max Mauro Dias Santos. On the applicability of an MILP solution for signal packing in CAN-FD. In *Proceedings of the IEEE 14th International Conference on Industrial Informatics (IEEE-INDIN)*, July, 2016. doi:10.1109/INDIN.2016.7819350.

17 Florian Polzlbauer, Iain Bate, and Eugen Brenner. Optimized frame packing for embedded systems. *IEEE Embedded Systems Letters*, 4(3):65–68, 2012. doi:10.1109/LES.2012.2208094.

18 Paul Pop, Petru Eles, and Zebo Peng. Schedulability-driven frame packing for multicluster distributed embedded systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 4(1):112–140, 2005. doi:10.1145/1053271.1053276.

19 Rishi Saket and Nicolas Navet. Frame packing algorithms for automotive applications. *Journal of Embedded Computing*, 2(1):93–102, 2006. doi:10.1.1.5.1953.

20 Kristian Sandstrom, C Norstom, and Magnus Ahlmark. Frame packing in real-time communication. In *Proceedings of the Seventh International Conference on Real-Time Computing Systems and Applications*, pages 399–403. IEEE, 2000. doi:10.1109/RTCSA.2000.896418.

21 Bogdan Tanasa, Unmesh Dutta Bordoloi, Petru Eles, and Zebo Peng. Reliability-aware frame packing for the static segment of FlexRay. In *Proceedings of the Ninth ACM international conference on Embedded software*, pages 175–184. ACM, 2011. doi:10.1145/2038642.2038670.

22 KW Tindell, Hans Hansson, and Andy J Wellings. Analysing real-time communications: controller area network (CAN). In *Proceedings of Real-Time Systems Symposium*, pages 259–263. IEEE, 1994. doi:10.1109/REAL.1994.342710.

23 Gökhan Urul. *A Frame Packing Method to Improve the Schedulability of CAN and CAN-FD*. PhD thesis, Middle East Technical University, Turkey, 2015.

24 Haibo Zeng, Marco Di Natale, Arkadeb Ghosal, and Alberto Sangiovanni-Vincentelli. Schedule optimization of time-triggered systems communicating over the FlexRay static segment. *IEEE Transactions on Industrial Informatics*, 7(1):1–17, 2011. doi:10.1109/TII.2010.2089465.

25 Wei Zheng, Qi Zhu, Marco Di Natale, and Alberto Sangiovanni Vincentelli. Definition of task allocation and priority assignment in hard real-time distributed systems. In *Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International*, pages 161–170. IEEE, 2007. doi:10.1109/RTSS.2007.40.

26 Qi Zhu, Yang Yang, Eelco Scholte, Marco Di Natale, and Alberto Sangiovanni-Vincentelli. Optimizing extensibility in hard real-time distributed systems. In *Real-Time and Embedded Technology and Applications Symposium, 2009. RTAS 2009. 15th IEEE*, pages 275–284. IEEE, 2009. doi:10.1109/RTAS.2009.37.