Frame packing algorithms for automotive applications

Rishi Saket, Nicolas Navet

Frame Packing Algorithms for Automotive Applications

Rishi Saket [1] - Nicolas Navet [2,*]

[1] Georgia Institute of Technology
College of Computing
Atlanta GA 30332
USA

[2] LORIA - INRIA
Campus Scientifique - BP 236
54506 Vandoeuvre-lès-Nancy
France

*Corresponding author. Tel.: +33 3 83581763; Fax: +33 3 83581701; E-mail: nnavet@loria.fr

Abstract

The set of frames exchanged in automotive applications must meet two constraints: it has to be feasible from a schedulability point of view and it should minimize the network bandwidth consumption. This latter point is important since it allows the use of low cost electronic components and it facilitates an incremental design process. The purpose of this study is to propose efficient algorithms for solving the NP-hard problem of generating a set of schedulable frames that minimize the bandwidth usage. This study presents novel algorithms for building bandwidth-minimizing sets of frames that meet the schedulability requirement. In our experiments, these proposals have proved to be more effective than the existing approaches.

keywords: Embedded systems, scheduling algorithms, Controller Area Network, heuristics, bandwidth-minimization problems

1 Introduction

Application processes in the different Electronic Control Units (ECUs) of a vehicle send and receive elementary pieces of information, that are called signals, over the network. Typical examples of such signals are the Revolution Per Minute (RPM) value and the speed of the vehicle. Signals have usually a small size and several of them can be transmitted in the same frame so as to minimize bandwidth consumption. In-vehicle applications are subject to stringent time constraints and most signals have a limited lifetime. This study proposes algorithms for building off-line the set of frames in automotive communications with the two objectives of minimizing the bandwidth consumption and ensuring the respect of timing constraints. At run-time, the packing of the frames according to the configuration and their transmission at the right point of time is usually performed by a middleware layer that masks the communication system services to producer and consumer processes.

Knowing the set of ECUs and for each ECU the set of signals that are to be sent over the network, their size, their deadline and their production period, the problem consists in building the set of the frames. This comes to define the set of signals composing each frame as well as to decide their characteristics of emission. In the case of a priority based Medium Access Control (MAC) protocol such as the Controller Areal Network (CAN), which is a de-facto standard in automotive applications [10], the priority of each frame has to be chosen and those priority choices will clearly have an impact on the respect of the deadlines.

In addition to schedulability, the set of frames must be constructed with the objective of minimizing the bandwidth consumption. The first reason lies on the fact that the possibility of adding new functionalities under the form of one or several additional ECUs must be preserved knowing that adding more ECUs implies that more signals will be exchanged on the bus. Such an incremental design is a standard procedure in the automotive industry. The second reason to minimize the bandwidth consumption is to allow the use of low power pro-

cessors, which are less expensive, and to decrease the frequency of transmission errors due to EMI (Electro-Magnetic Interferences) if a lower network data rate is possible. The problem to solve is to find a configuration of frames under schedulability and bandwidth minimization constraints. This problem is known to be NP-hard (see [11] for a proof) and, as it will be shown, it cannot be solved using an exhaustive approach even for a small number of signals and/or ECUs. The solution is thus to find efficient heuristics.

The frame packing problem has not been extensively studied in the literature and particularly when real-time constraints are matter of concern. In [11], several heuristics are presented to build a set of frames exchanged over a CAN network in such a way as to minimize the bandwidth consumption but without considering the schedulability of the solution. The configuration tools of the middleware Volcano (see [3]) provides solutions that takes account of the feasibility but the algorithms implemented by this commercial product are not published. The frame packing problem has been recently studied in [13] for distributed systems made of both time-triggered and event-triggered networks interconnected via gateways but bandwidth consumption was not an objective to achieve. To our best knowledge, the only publically available study presenting solutions to our problem in the context of priority based MAC protocols is [7] where two complementary strategies have been proposed. The first one, termed Bandwidth-Best-Fit decreasing (BBFd), can be applied to large sized problems in the context of in-vehicle applications (several hundreds of signals). The second one, termed the "semi-exhaustive" heuristic (SE) has proved to be slightly more efficient in the experiments but its use is limited to small size problems (less than 12 signals emitted by each stations). The first algorithm is inspired from "bin-packing" algorithms (see [4] for a survey) while the second one is an exhaustive search through the solution's space where one would cut some branches that are judged not promising.

In this paper, two algorithms are proposed. The first one is a greedy algorithm for building the set of frames starting from the set of signals. Its com-

plexity is compatible with large size problems. The second algorithm takes as input an unfeasible set of messages and tries to transform it into a feasible one. The strategy is to lower the deadline constraints of the frames by isolating the most demanding signals. Another contribution of the paper is a proof that the Audsley algorithm can be used for optimally allocating priorities in the context of priority based MAC if the blocking factor (see paragraph 2.2) is the same for the whole message set.

In this study, the performance metrics are the network bandwidth consumption and the capability to find schedulable solutions. Our proposals will be evaluated by comparison with the algorithm presented in [7] wich has proved to be much more effective than 'bin-packing' heuristics and than a naive strategy consisting in having one signal per frame.

Section 2 is devoted to a formal description of the problem, the assumptions and notations are given and the complexity of an exhaustive approach is derived. In the same section, we also explain how to set the priorities of the frames in an optimal way and prove the correctness of the approach. In section 3, the Bi-Directional Frequency Fit algorithm (BDFF) is described and its performance are assessed. Finally, in section 4, a new decomposition strategy is proposed for transforming a non-feasible set of messages into a feasible one.

2 Problem Statement

The problem is to construct a set of frames $F = \{f_1, f_2, ..., f_k\}$ from a given set of signals $S = \{s_1, s_2, ..., s_n\}$ such that the resulting set of frames on each station is schedulable, which means that none of the signals transmitted miss its deadline, while utilizing as little bandwidth as possible. The location of application processes is fixed so each signal is associated to the station that produces it. Each signal s_i has a tuple consisting of 4 values: (N_i,T_i,C_i,D_i) where

- N_i denotes the station containing the signal,

- T_i is the period of production of the signal at that station. In practice the clocks of the stations are not synchronized but we assume that the first production of all signals located on the same ECU takes place at the same time,

- C_i is the size of of the signals in bits. It is assumed that C_i is smaller than the maximum data size of the frame (segmentation issues are not taken into account),

- D_i is the deadline relative to the production time of the signal s_i, it is the maximum duration between the production of the signal on the sender side and its reception by all consumers. In this paper we assume that the deadline of a signal is equal to the production period ($D_i = T_i$) but the proposed strategies are still valid for deadlines smaller than the period.

The communication network is a priority bus that might be CAN [5], VAN [6] or the J1850 [15]. In the following, the numerical results are obtained with a CAN network at 500kbits/s. The standard CAN data frame (CAN 2.0A, see Figure 1) can contain up to 8 bytes of data for an overall size of, at most, 135bits, including all the protocol overheads. The sections of the frames are:

- the header field, which contains the identifier of the frame, the Remote Transmission Request bit that distinguishes between data frame (RTR set to 0) and data request frame (RTR set to 1) and the Data Length Code (DLC) used to inform of the number of bytes of the data field,

- the data field having a maximum length of 8 bytes,

- the 15 bit Cyclic Redundancy Check (CRC) field which ensures the integrity of the data transmitted,

- the Acknowledgment field (Ack). On CAN, the acknowledgment scheme solely enables the sender to know that at least one station, but not necessarily the intended recipient, has received the frame correctly,

- the End-of-Frame (EOF) field and the intermission frame space which is the minimum number of bits separating consecutive messages.

The exact size of the CAN frame overhead is dependent of the data because of the 'bit-stuffing' mechanism[1]. Each CAN frame contains up to 8 data bytes and we consider an overhead of 64 bits, which is a reasonable value.

Each frame resulting from the packing will contain a given set of signals that do not change over the successive transmissions of the frame. A frame is characterized by the tuple $(N_i^{'},T_i^{'},C_i^{'},D_i^{'})$ where,

- $N_i^{'}$ denotes the station that sends the frame.
- $T_i^{'}$ is the frame transmission period at that station. This period is the smallest of the periods of the signals contained in the frame.
- $C_i^{'}$ is the size of the frame in bits. It is the sum of all the signal sizes and a constant overhead fixed to 64 bits.
- $D_i^{'}$ is the deadline of the frame which depends on the signals composing the frame,
- $P_i^{'}$ is the priority of the frame for the MAC protocol.

It is usual in automotive networks that a part of the traffic has no real-time constraints (for instance, diagnosis frames) and such frames are assigned a priority lower than all real-time frames. Without further knowledge off the application, it is assumed that these frames have a size of 128 bits (8 bytes of data plus the overhead), which will be the blocking factor for all real-time frames (ie. the maximum time interval during which one frame can be delayed by a lower priority frame).

[1] Bit stuffing is an encoding method that enables resynchronization when using Non-Return-to-Zero (NRZ) bit representation where the signal level on the bus can remain constant over a longer period of time (e.g. transmission of '000000..'). Edges are generated into the outgoing bit stream in such a way to avoid the transmission of more than a maximum number of consecutive equal-level bits (5 for CAN).

2.1 Problem complexity

The problem of building frames from signals is similar to the 'bin-packing' problem and it was proven to be NP-hard in [11]. Nevertheless on small size problems an exhaustive approach might be used. One thus has to determine the exact complexity of our specific problem.

To group a set of n signals in k non-empty frames comes to create all the partitions of size k from the set of signals. The complexity of this problem is known, it is the Stirling number of the second kind (see [1] page 824):

$$\frac{1}{k!}\sum_{i=0}^{k}(-1)^{(k-i)}\binom{k}{i}i^n$$

The number k of frames per station can vary from 1 to n where n is the number of signals produced by the station. The number of frames to envisage on a station i that produces n signals is thus:

$$\mathcal{S}_i = \sum_{k=1}^{n}\frac{1}{k!}\sum_{i=0}^{k}(-1)^{(k-i)}\binom{k}{i}i^n$$

For example, three signals a, b, and c on a same station leads to five different partitions : $[(a,b,c)]$, $[(a),(b,c)]$, $[(a,b),(c)]$, $[(a,c),(b)]$ and $[(a),(b),(c)]$.

If we consider a set of m stations, the solutions space becomes $\prod_{i=1..m} S_i$ where S_i is the number of possible frames for station i. The solutions space grows very quickly as soon as n and m increase. For instance, with 10 signals per station ($S_i = 115975$) and 5 stations, an exhaustive search would need to consider about $2 \cdot 10^{25}$ cases. Moreover, this evaluation did not take account of the determination of frame priority. One can see that for real industrial cases an exhaustive approach cannot be applied. This justifies the design of specific heuristics.

2.2 Optimal priority allocation

Under the Fixed Priority Preemptive (FPP) policy, the response time of a task only depends on the set of higher priority tasks and not on the relative priorities between these higher priority tasks. Starting from this observation, Audsley proposed in [2] a priority allocation algorithm that runs in $O(n^2)$ where n is the number of tasks. This algorithm is optimal: if a solution exists then it will necessarily be found by the algorithm. The strategy is to start from the lowest priority level (m) and to search a task that is feasible at this priority level. In case of failure, one can conclude that the set of tasks is not schedulable. The first feasible task at level m is assigned to that priority. Once the priority level m is given to a task, the algorithm tries to find a feasible task at level $m-1$ and so on until priority 1 which is the highest priority of the system.

In the general case this algorithm cannot be applied to message scheduling under the Non-Preemptive Fixed Priority (NPFP) policy which is the medium access algorithm for priority buses. Indeed, on a network the response time of a frame not only depends on the higher priority frames but also on the set of lower priority frames because of the blocking factor (maximal time interval during which one frame can be delayed by a lower priority frame). In our particular context the blocking factor is equal for all frames since we assume the existence of a non real-time traffic (e.g. diagnostic frames). Without any other assumptions on this traffic, one must consider the blocking factor as being the size of the largest frame compliant with the communication protocol. In this case, the conditions are fulfilled to apply the Audsley algorithm in order to evaluate the schedulability of a set of frames. The proof that the Audsley algorithm is optimal in our context in given in Appendix B.

3 The Bi-directional frequency fit heuristic

The name of this algorithm has been given by analogy with the terminology used in 'bin-packing' problems with off-line resolution (*Best-Fit decreasing - First-*

Fit decreasing, see [4]). In 'bin-packing' problems, the objective is to minimize the number of boxes (of frames here). In our context the goal is to minimize the network bandwidth consumption without taking account of the number of frames.

3.1 Rationale of the algorithm

The motivation for proposing a new algorithm for this problem comes from the performance analysis of the Bandwidth-Best-Fit decreasing (BBFd) which, in [7], proves to be much more effective than other a priori possible strategies ("one signal per frame", "First-Fit Decreasing" - FFD, "Best-Fit Decreasing" - BFD) and not far away from an exhaustive search when this is possible. The first step of BBFd is to sort all signals in decreasing order of bandwidth consumption on each station. Then, starting from the beginning of the sorted array of signals, each signal is inserted in the frame that minimizes the bandwidth usage and ensures feasibility. However this strategy is not always effective. Bin-packing is started from the beginning of the array, maintaining a set of frames that are not fully packed at each step. Now as one progresses towards the end of the array, signals with lower frequencies tend to be encountered since signals are sorted according to their bandwidth usage. These signals may be packed into an incompletely packed frame with high frequency signals. This will result in a waste of bandwidth because the frame has a high frequency itself and there is an increase in its size. In our experiments with uni-directional heuristics such as BBFd but also FFD and BFD, we observed a large number of such anomalies.

The underlying idea of the "Bi-directional Frequency Fit" (BDFF) heuristic is that the signals with the same frequencies should be grouped together as this reduces the bandwidth consumption. At each station the heuristic sorts the signals in decreasing order of their frequency. It creates an empty frame, adds it to a $front\_set$, and starts packing signals into it from the beginning of the array until a new frame is needed to be created. At this point, it switches to the end of the array, creates a new frame, adds it to a $back\_set$ and starts

packing signals till a new frame needs to created and so on. Only frames from the $front\_set$ are used to pack signals from the beginning and only frames from the $back\_set$ for packing signals from the end of the array. This continues till all signals are packed. In this way, by maintaining two sets of frames, the heuristic tries to ensure that lower frequency signals are not packed in higher frequency frames. The feasibility of the set of frames thus composed is tested by the Audsley algorithm as explained in subsection 2.2.

3.2 Algorithmic description

In the following, F is the set of all frames, F^i_{front}, F^i_{back} respectively denotes the $front\_set$ and $back\_set$ at station i while S_j is the bi-directional list of signals at station j sorted in decreasing order of frequency. The algorithm of the "Bi-directional Frequency Fit (BDFF)" heuristic is given below:

1. Construct $F = \{\}$

2. At each station j do the following:

   2.a Sort the signals in decreasing order of frequency in a bi-directional list S_j.

   2.b Construct $F^j_{front} = F^j_{back} = \{\}$. Set $Front = true$.

   2.c If S_j is empty do $F = F^j_{front} \cup F^j_{back} \cup F$ and proceed to the next station and goto step 2, and if all the stations are done goto step 3.

   2.d If $Front = true$ goto 2.d.1 else goto 2.d.4.

      2.d.1 Remove a signal s_i from the front of the list S_j and construct a new frame f_{new} and insert s_i in it and add f_{new} to F^j_{front}.

      2.d.2 If S_j is empty goto step 2.c.

      2.d.3 Look at the signal s at the front of the list S_j. Find the frame f in F^j_{front} which mini-

mizes the bandwidth utilization with a positive deadline (see Appendix A) and which does not exceed the maximum data size if s is inserted in it. Compare this bandwidth to the one obtained by inserting s in a new frame. If there is no frame f in F^j_{front} which can accommodate s or if the bandwidth is minimized by inserting s in a new frame then set $Front = false$ and goto step 2.c; else remove the signal s from the front of the list S_j and add it to the frame f and update the characteristics of f and goto step 2.d.2.

2.d.4 Remove a signal s_i from the back of the list S_j and construct a new frame f_{new} and insert s_i in it and add f_{new} to F^j_{back}.

2.d.5 If S_j is empty goto step 2.c.

2.d.6 Look at the signal s at the back of the list S_j. Find the frame f in F^j_{back} which minimizes the bandwidth utilization with a positive deadline (see Appendix A) and which does not exceed the maximum data size if the s is inserted in it. Compare this bandwidth to the one obtained by inserting s in a new frame. If there is no frame f in F^j_{back} which can accommodate s or if the bandwidth is minimized by inserting s in a new frame then set $Front = true$ and goto step 2.c; else remove the signal s from the back of the list $S_j$and add it to the frame f and update the characteristics of f and goto step 2.d.5.

3. Feasibility test of the set of frames with the Audsley algorithm:

   3.a If the set F is feasible return SUCCESS otherwise go to step 3.b

   3.b Construct the set of frames $F' \subseteq F$, for which no priority has been found.

      3.b.1 If all the frames in F' contain one signal each return FAILURE else goto step 3.b.2

      3.b.2 Find the frame in F' containing at least 2 signals and which has the least difference between the worst case response time and the deadline at the lowest priority which has not been assigned. Let the frame be f'.

      3.b.3 Remove the s' signal from f' with the smallest deadline, and make a new frame f'' containing only s'. Update the characteristics of f' and add f'' to the set F. Goto step 3.a.

This heuristic is composed of two distinct parts. The first parts (step 1 and 2) aims at constructing a solution that minimizes the bandwidth consumption. The second part (step 3) deals with the feasibility of the proposed solution that is tested with the Audsley algorithm (see paragraph 2.2). Should the Audsley test fail, then the heuristic tries to reduce the deadline constraints of the frames by isolating the most demanding signals. The frame initially chosen for the decomposition is the one that exceeds the least its deadline at the lowest priority level that has not been successfully assigned. It is thus the frame that is the more "likely" to respect its deadline if it should contain less data. To determine which frame to decompose, it is necessary to compute a response time for each frame for which a priority has not been found: one gives the considered frame the first priority not assigned by the Audsley algorithm. The higher priority frames stay unchanged and the lower priority frames have no influence at all.

This decomposition scheme is the one presented in [7], which will enable a fair comparison between BDFF and BBFd. In Section 4, we propose a new strategy that outperforms this one.

At each station j, the construction of the set $F^j_{front} \cup F^j_{back}$ takes $O(n_j^2)$ time where n_j is the number of signals on that station. The decomposition can take, in the worst case, $O(n_{total})$ decompositions and for each decomposition $O(n_{total}^2)$ response time calculations, where n_{total} is the total number of signals in all the stations. In our experiments made on industrial-scale problems (hundreds of signals), this complexity did not raise time problem since the average computation time of a solution with the largest network load considered (see §3.3) was less than one second on an Intel Centrino 1.7Ghz processor.

3.3 Performance evaluation

The BDFF heuristic is compared to the existing BBFd heuristic with regard to the bandwidth consumption of the resulting set of frames. The heuristics were implemented in C++ and the feasibility of the solutions was tested using the *rts* software [8] that implements the Audsley Algorithm as well as the worst-case response time computation on CAN. Only the results induced by configurations that are feasible with both strategies are taken into account since the use of non-feasible set of frames is ruled out in the context of in-vehicle applications. The experiments are performed on randomly generated sets of messages where the parameters, chosen so as to be realistic with typical in-vehicles applications (see [16, 9]), are the following:

1. the bandwidth of the CAN network is fixed at 500 kilobits/sec,
2. the size of a signal is given by a random variable uniformly distributed from 1 to 24 bits,
3. the signal period is given by a random variable uniformly distributed from 5 to 100 msec in steps of 5 msec,
4. the signal deadline is fixed equal to the signal period,

5. the maximum frame data size is 64 bits as defined by the CAN protocol [5],
6. the overhead per frame is 64 bits,
7. the number of stations is 1, 2, 5, 7, 10, 12 or 15,
8. the nominal load (i.e. is the load brought by the data alone) for the whole set of stations is either 10%, 15% or 20%.

Figure 2 shows the total network load for 150 configurations returning feasible solutions for both of the algorithms. In each pair of curves, one represents the load with BBFd and the other the load with BDFF at a particular nominal load. On the x axis, the number of stations varies.

The results of the experiments show that the BDFF heuristic outperforms the BBFd heuristic in terms of minimizing the bandwidth consumption whatever the nominal load and the number of stations. The gain reaches up to 21% (one station - 20% of nominal load) and it is especially important when the number of stations is smaller. As the number of stations increases the better performance of the BDFF heuristic becomes less evident.

A closer analysis of the data reveals that for BDFF the set of frames built after step 2 of the heuristic is more likely to need decomposition as the number of stations increases. It was also observed that in the subsequent decomposition procedure, in general, a large number of new frames were created and the bandwidth consumption of the resulting feasible configuration was significantly higher than the bandwidth obtained with the BBFd heuristic. This is due to the fact that the decomposition strategy does not always succeed in increasing the deadline of the decomposed frames effectively. This led us to propose a new decomposition scheme.

4 Deadline relaxing decomposition scheme

The decomposition scheme used up to now for evaluating BDFF against BBFd was proposed in [7]. First, this algorithm identifies the frame that exceeds its

deadline by the smallest amount of time when scheduled at the lowest priority level that has not been successfully assigned. Then, the signal contained in that frame with the smallest deadline is isolated in a newly created frame, increasing thus the deadline of the original frame.

However, this scheme often fails to substantially relax the original deadline because the deadline of a frame depends not only on the deadlines of the individual signals but also on their periods (see Appendix A). In addition, this scheme creates frames with just one signal and, thus, results in inefficient packing due to the protocol overhead. Moreover, in the common situation in which deadline is equal to period, these one-signal frames have a high frequency, thus greatly increasing bandwidth consumption. The goal of this section is to propose a decomposition scheme which is more effective in increasing the deadlines of the resulting frames while not significantly increasing the bandwidth consumption.

4.1 Algorithmic description

The idea of the algorithm is to choose the signals to remove from the original frame in such a manner that the deadline of the frame without these signals is maximum. Another frame is formed into which the chosen signals are added in such a way that the deadline of the new frame is always greater than or equal to the deadline of the original frame. If this strategy is successful, one ends up with two frames having deadlines no lesser than the original one and being thus more likely to be schedulable. The algorithm replaces the step 3.b.3 of the BDFF scheme described in paragraph 3.2:

3.b.3 The frame $f^{'}$ is to be split and F is the set of frames. Set $initial\_deadline$ to the deadline of $f^{'}$. Create $f_1 = f^{'}$ and an empty frame f_2. Set $continue = true$.

3.b.3.a If f_1 contains only one signal then goto step 3.b.3.f

3.b.3.b Choose s from f_1 such that

$$s = \text{argmax}_{s_i \in f_1}(deadline(f_1 - \{s_i\})$$

where $deadline(f)$ returns the deadline of frame f given the signals composing f.

3.b.3.c If $deadline(f_2 + \{s\}) < initial\_deadline$ then goto step 3.b.3.f

3.b.3.d If $deadline(f_1 - \{s\}) > initial\_deadline$ set $continue = false$.

3.b.3.e Set $f_1 = f_1 - \{s\}$ and $f_2 = f_2 + \{s\}$. If $continue = true$ goto step 3.b.3.a

3.b.3.f Remove $f^{'}$ from F. Add f_1 and f_2 to F.

4.2 Performance evaluation

The performance evaluation of the deadline relaxing decomposition scheme was performed considering configurations generated by both the BBFd heuristic and the BDFF heuristic. Only the frame sets which required decomposition due to unfeasibility were considered. The parameters of the evaluation were the same as for the experiment of paragraph 3.3 except that the number of stations was fixed at 10 and only high nominal loads were considered. The performance of the deadline relaxing decomposition scheme was assessed against the previous decomposition scheme. At each load level, both algorithms were executed on 100 configurations requiring decomposition.

Tables 1 and 2 show the performance in terms of bandwidth and feasibility of the deadline relaxing decomposition scheme (denoted D2) against the old scheme (D1) on unfeasible frame sets created by BBFd and BDFF. For the bandwidth consumption only those configurations were considered which resulted in success for both the decomposition schemes. Whatever the load and the decomposition algorithm, D2 clearly outperforms D1 in terms of bandwidth.

Regarding feasibility, D2 also proves to behave much better than D1. For instance at 25% nominal, there are approximatively 3 times more feasible configurations with D2 than with D1. An important observation is that on our experiments it never happens that a configuration became feasible with D1 and not with D2.

From table 1 and 2, one sees that the overall efficiency of BDFF plus D2 in terms of bandwidth and feasibility is clearly better than the one from BDFd plus D1 which was the solution proposed in [7]. However, with a same decomposition scheme, BBFd tends to be better than BDFF in terms of feasibility which means that BBFd must not be ruled out especially on highly loaded systems.

5 Conclusion

In this study, algorithms for solving the problem of building feasible sets of frames that minimize the bandwidth consumption have been proposed. The proposals have proved to be more effective than existing solutions published in [7] both in terms of bandwidth consumption and capability of finding feasible sets of messages.

The BDFF heuristic can be used as a starting point for other optimization algorithms to direct the search towards promising parts of the solution's space. In particular they might be included in the initial population of a genetic algorithm, the initial population having in general a strong impact on the performance of the algorithm (see for instance [18]). One can additionally try to improve the solution given by BDFF in terms of bandwidth usage with a local optimization procedure as it is classically done in optimization (see for instance [12]). A simple scheme that tries to permutes pairs of signals has been proposed in [7]: it brought small improvements (about 0.5% in bandwidth) and better heuristics remain to be found.

In this study, frames are formed by statically binding each signal to a frame. The frames are then periodically transmitted even if some signals having lower

frequencies than the frame have not been generated. This leads to a waste of bandwidth but on the other hand the memory needed to store the characteristics of the frames is minimum. Another possibility to solve the frame packing problem is to build a set of frames such that each frame only contains signals that have been produced at the transmission time of the frame. However, it has to be pointed out that this scheme is restricted to specific applications, in particular having a small LCM of the signal periods, since the number of frames becomes in general too important for being stored especially in the context of automotive systems. Experiments conducted with a simple First Fit scheme (a frame is made of a number of signals whose production dates occurs consecutively) suggest to us that the gain in bandwidth consumption is very important (up to 30% over BDFF - only feasible set of frames were considered). A future work consists in conceiving algorithms for this problem with a reasonable algorithmic complexity and that perform well in terms of bandwidth and feasibility.

Acknowledgement: The authors would like to thank Jörn Migge from PSA Peugeot Citroën company for providing the *rts* software [8] and for some helpful comments on this study.

References

[1] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions.* Dover Publications, ISBN 0-486-61272-4, 1970.

[2] N. C. Audsley. Optimal priority assignment and feasibility of static priority tasks with arbitrary start times. Technical Report YCS164, University of York, November 1991.

[3] L. Casparsson, A. Rajnák, K. Tindell, and P. Malmberg. Volcano - a revolution in on-board communications. Technical report, Volvo Technology Report, 1999.

[4] E. G. Coffman, M. R. Garey, and D. S. Johnson. *Approximation Algorithms for NP-Hard Problems*, chapter Approximation Algorithms for Bin Packing: a Survey. PWS Publishing Company, 1996.

[5] ISO. ISO International Standard 11898 - Road vehicles - Interchange of digital information - Controller Area Network (CAN) for high-speed communication, 1993.

[6] ISO. ISO International Standard 11519-3 - Road vehicles - Low-speed serial data communication - Vehicle Area Network (VAN), 1994.

[7] R. Marques, N. Navet, and F. Simonot-lion. Frame packing under real-time constraints. In *Proceedings of the 5th IFAC International Conference on Fieldbus Systems and their Applications (FeT'2003)*, 2003.

[8] J. Migge. RTS - a tool for computing response time bounds, 2002. Program and manual available at `http://www.loria.fr/~nnavet`.

[9] N. Navet, Y.-Q. Song, and F. Simonot. Worst-case deadline failure probability in real-time applications distributed over CAN (Controller Area Network). *Journal of Systems Architecture*, 46(7):607–618, 2000.

[10] N. Navet, Y.-Q. Song, F. Simonot-Lion, and C. Wilwert. Trends in automotive communication systems. *Proceedings of the IEEE, special issue on Industrial Communications Systems*, 96(6):1204–1223, June 2005.

[11] C. Norström, K. Sandström, and M. Ahlmark. Frame packing in real-time communication. In *Proceedings of the Seventh International Conference on Real-Time Systems and Applications (RTCSA'00)*, 2000. Extended version available as Mälardalen Real-Time Research Center Technical Report.

[12] T. Osogami and H. Okano. Local search algorithms for the bin packing problem and their relationships to various construction heuristics. In *Proceedings of IPSJ SIGAL*, 1999. Also available as IPSJ Technical Report number 99-AL-69-5.

[13] P. Pop, P. Eles, and Z. Peng. Schedulability-driven frame packing for multi-cluster distributed embedded systems. In *Proceedings of the 2003 ACM SIGPLAN conference on Language, compiler, and tool for embedded systems*, 2003.

[14] G. Quan and X. Hu. Enhanced fixed-priority scheduling with (m,k)-firm guarantee. In *Proceedings IEEE Real-Time System Symposium (RTSS'2000)*, 2000.

[15] Society of Automotive Engineers. Class B data communications network interface - SAE J1850 standard - rev. nov96, 1996.

[16] K. Tindell and A. Burns. Guaranteed message latencies for distributed safety-critical hard real-time control networks. Technical report, Department of Computer Science, Univ. of York (UK), May 1994. Technical Report YCS229.

[17] K. Tindell, A. Burns, and A.J. Wellings. Calculating Controller Area Network (CAN) message response times. *Control Eng. Practice*, 3(8):1163–1169, 1995.

[18] C. H. Westerberg and J. Levine. Investigation of different seeding strategies in a genetic planner. In Springer-Verlag, editor, *Proceedings EvoWorkshops2001: EvoCOP, EvoFlight, EvoIASP, EvoLearn, and EvoSTIM*, LNCS 2037, 2001.

A Deadline of a frame after the addition of a signal

The transmission period of a frame containing several signals is the smallest production period of the signals and the transmission of the frame will be synchronized with the production of the signal having the smallest period. On the other hand, the deadline of the frame is not the smallest deadline due to possible

offsets between the production dates of the signals and the actual transmission dates of the frame. Let us consider the example shown on figure 3 with two signals s_1 and s_2 having respectively a period $T_1 = 10$ and $T_2 = 14$, and a deadline (relatively to the production date) $D_1 = 10$ and $D_2 = 14$. The signal s_2 produced at time 14 is actually sent at time 20 and the deadline of the frame must thus be equal to 8 in order to respect the timing constraint of s_2. One can also note that the transmissions made at times 10 and 40 do not include any new value for s_2. In practice, the value of s_2 might be re-transmitted or not but there will not be any timing constraint on the frame induced by signal s_2.

To determine the deadline of the frame, one must find the largest offset between a production date and the transmission of the next frame. One wants to include signal s_i in frame f_k already containing the signals $s_1^k, s_2^k, ..s_n^k$. One denotes s_{min} the signal with the smallest period of the set $\{s_1^k, s_2^k, ..s_n^k\} \cup s_i$, the period of f_k becomes $T_k^* = T_{min}$. The relative deadline of f_k is $D_k^* = \min\{D_j - \text{Offset}(T_{min}, T_j) \,|\, s_j \in \{s_1^k, s_2^k, ..s_n^k\} \cup s_i\}$ where $\text{Offset}(a, b)$ returns the largest possible duration between the production date of a signal with period $b \geq a$ and the transmission of the frame of period a that contains the signal. It has been shown in [14] that $\forall k_1, k_2 \in \mathbb{N}\ k_1 \cdot a - k_2 \cdot b = q \cdot \gcd(a, b)$ with $q \in \mathbb{Z}$. In our context, one imposes $a > k_1 \cdot a - k_2 \cdot b \geq 0$ and thus $\text{Offset}(a, b) = (\frac{a}{gcd(a,b)} - 1) \cdot gcd(a, b) = a - gcd(a, b)$. In our example, the deadline must be set to 6.

B Audsley Algorithm for Non-Preemptive fixed-Priority Scheduling

In this section, we give a proof that the Audsley algorithm is optimal, in the sense that if a feasible priority allocation exists then it will necessarily be found by the algorithm, for Non-Preemptive Fixed-Priority (NPFP) scheduling if the

blocking factor is the same for all frames of the message set. In our case, we assume the existence of at least one non real-time frame whose size if greater than or equal to the biggest frame of the real-time message set. In the automotive context, such frames usually exist for instance for exchanging diagnosis data among ECU's.

For a given set of frames F of cardinality m there exists a set of feasible priority allocation denoted by $\mathcal{A}$. One denotes $\gamma(k)$ the index of the task having the priority level k under a given priority allocation. The notation $[\gamma(m), ..., \gamma(k)]$ indicates a partial priority allocation where only the priority levels from m to k have been fixed. One calls property 1 the fact that under NPFP, the worst-case response time of a task at level k only depends on the workload induced by tasks from priority 1 to k and on the blocking factor (see, for instance, [17]). Property 2 is the existence of a single blocking factor for the whole task set.

The Audsley Algorithm (AA) searches for each priority level a task that is feasible at this priority. It starts from the lowest priority level m and runs down to the highest level which is 1. The proof is made by induction on the priority levels. The induction hypothesis being that, up to step k, the Audsley algorithm has found a partial priority allocation such that there necessarily exists at least one allocation in $\mathcal{A}$ which contains the partial allocation obtained by the algorithm up to step k. We assume $\mathcal{A} \neq \emptyset$.

For priority level m, AA returns $\gamma(m)$ such that frame $f_{\gamma(m)}$ is feasible at level m (there exists at least a frame feasible at level m thus AA by trying the possibilities one by one will find one). Assume that none of the allocation in $\mathcal{A}$ has chosen $\gamma(m)$ for level m. Let a be one allocation in $\mathcal{A}$, thus $\gamma^a(m) \neq \gamma(m)$. Under a, $f_{\gamma(m)}$ is thus assigned a priority $n < m$. One can exchange the priority of frames $f_{\gamma^a(m)}$ and $f_{\gamma^a(n)}$ under a while keeping feasibility since $f_{\gamma^a(n)}$ $(=f_{\gamma(m)})$ is feasible at level m and the total workload at priority n is lower than or equal to the workload at level m and because of property 1 and 2. Thus, there necessarily exists at least one feasible allocation with the choice made by the Audsley algorithm for level m.

Priorities for levels m to k have been chosen with AA and one obtains the partial priority allocation $[\gamma(m), ..., \gamma(k)]$ with $[\gamma(m), ..., \gamma(k)]$ belonging to at least one feasible solution (induction hypothesis). For level $k-1$, AA returns the frame with index $\gamma(k-1)$ which is feasible at this priority level. As for the case of priority m, since there exists at least a frame feasible at level $k-1$, AA by trying the possibilities one by one will find one. Assume that none of the allocation in $\mathcal{A}$ has chosen $\gamma(k-1)$ for level $k-1$. Let a be an allocation in $\mathcal{A}$ that contains $[\gamma(m), ..., \gamma(k)]$ with $\gamma^a(k-1) \neq \gamma(k-1)$. Using the same argument as for level m, one shows that a will also be feasible with $\gamma^a(k-1) = \gamma(k-1)$. Thus, AA always returns an allocation belonging to $\mathcal{A}$.

Tables

Nominal load	Number of successes with D1	Number of successes with D2	Av. bandwidth usage with D1 (%)	Av. bandwidth usage with D2 (%)
20%	96	100	71.8	70.4
22.5%	61	89	78.2	76.9
25%	16	52	84.4	82.3

Table 1: Efficiency of the new decomposition scheme (D2) against the existing one (D1) on message sets generated with BBFd.

Nominal load	Number of successes with D1	Number of successes with D2	Av. bandwidth usage with D1 (%)	Av. bandwidth usage with D2 (%)
20%	100	100	70.6	67.6
22.5%	68	95	77.7	73.8
25%	13	37	83.6	79.2

Table 2: Efficiency of the new decomposition scheme (D2) against the existing one (D1) on message sets generated with BDFF.

Figure Captions

Figure 1. Format of the CAN 2.0A data frame.

Figure 2. Total network load for BDFF and BBFd with a number of stations varying from 1 to 15 and a nominal load of 10, 15 and 20%.

Figure 3. Two signals with production periods equal to 10 and 14. The signals are transmitted in a frame synchronized with the signal having the smallest period. The dotted line indicates when the signal with period 14 is actually transmitted.

Figures

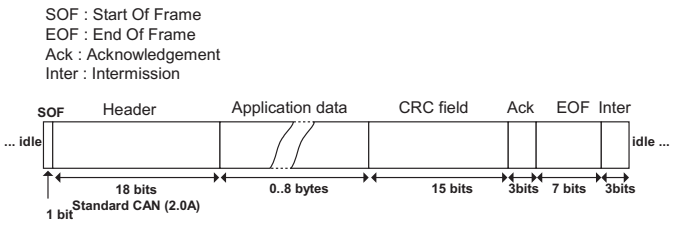

Figure 1: Format of the CAN 2.0A data frame.

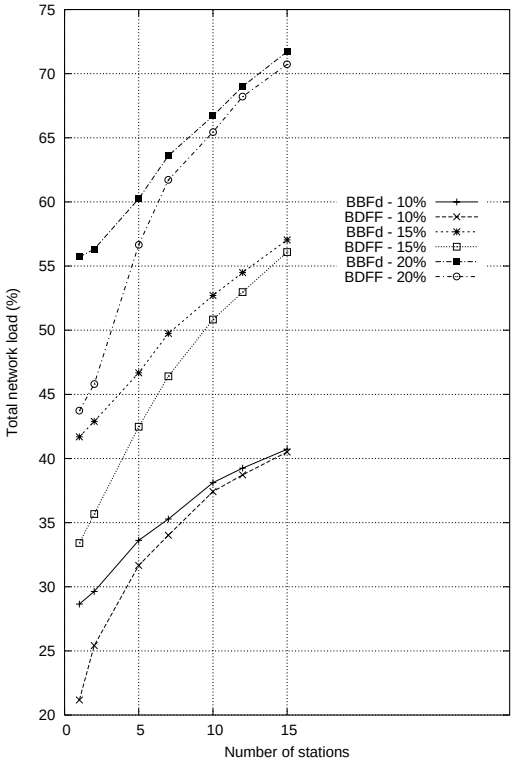

Figure 2: Total network load for BDFF and BBFd with a number of stations varying from 1 to 15 and a nominal load of 10, 15 and 20%.

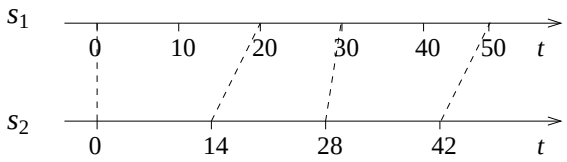

Figure 3: Two signals with production periods equal to 10 and 14. The signals are transmitted in a frame synchronized with the signal having the smallest period. The dotted line indicates when the signal with period 14 is actually transmitted.