Optimised Frame Packing for Embedded Systems

Florian Pölzlbauer (Student Member, IEEE), Iain Bate (Member, IEEE), Eugen Brenner

***Abstract*—During system synthesis (e.g. task allocation) the transmission of messages between tasks is usually addressed in a simplistic way. If a message is exchanged via an external bus, it is assumed each message is packed in an individual frame. This assumption leads to an over-estimation of bus bandwidth demand and frame response time. For some systems this pessimism is not acceptable (e.g. automotive), therefore frame packing is often performed where multiple messages are packed into a single frame. In this paper an improved frame packing approach is provided.**

***Keywords*-real-time systems; frame packing; minimize bandwidth demand; heuristic; automotive; bin packing;**

I. Introduction

In the literature the Task Allocation Problem (TAP) has been addressed many times. Although the approaches differ in several aspects, one common issue can be identified: If a message (that is exchanged between tasks) is sent via an external serial bus, it is assumed that the message is either transmitted in a separate frame (message.size $\leq$ frame.payload.max) or split into several frames (message.size $>$ frame.payload.max) and joined to one message again at the receiving processing node. For embedded real-time systems, the schedulability (i.e. ability to meet the system's timing requirements) is of upmost importance without compromising predictability. The simplistic assumption can lead to system configurations that are deemed to be unschedulable, and are usually rejected or punished by the TAP optimisation algorithm (since in general only feasible configurations are of interest). To overcome these issues, several messages must be packed in a single bus frame. In the literature, this problem is known as the Frame Packing Problem (FPP). In this paper we improve on state-of-the-art approaches.

This work is structured as follows: First we define the frame packing problem and provide an overview of state-of-the-art approaches. Then we derive optimality criteria for frame packing decision making. Based on these criteria we present an optimised frame packing heuristic. Later, we present experimental results that show the efficiency of our approach. Finally we draw our conclusions and identify future research aspects.

Symbol	Description	Symbol	Description
m	message	s_m	data size of message
f	frame	pay_f	payload of frame
M	set of messages	pm	max. payload of frame
F	set of frames	oh_f	overhead of frame
T	period	bw	bandwidth demand
D	deadline	br	baudrate of bus

Manuscript received November 04, 2011; revised March 09, 2012; accepted May 23, 2012. Date of publication MMM DD, 2012. Date of current version MMM DD, 2012. This manuscript was recommended for publication by NNN.

The authors are with Virtual Vehicle, University of York - Department of Computer Science, Graz University of Technology - Institute for Technical Informatics. (e-mail: florian.poelzlbauer@v2c2.at; iain.bate@cs.york.ac.uk; brenner@tugraz.at).

Digital Object Identifier XXX

II. Frame packing

The FPP is defined as follows: A set of messages $M = \{m_1, m_2, \ldots, m_n\}$ must be packed into a set of bus frames $F = \{f_1, f_2, \ldots, f_k\}$, subject to the constraint that the set of messages in any frame fits that frame s max. payload. Usually, the FPP is stated as an optimisation problem. The most common optimisation objectives are: 1) minimise the number of needed frames; or 2) maximise the schedulability of the resulting frame-set. A message is defined by $m_i = [s_i, T_i, D_i]$. A frame is define by $f_j = [pm_j, M_j, T_j, D_j]$. In general each frame may have its individual max. payload (depending on the bus protocol). However, usually all frames on the same bus have the same max. payload.

The FPP can be seen as a special case of the Bin Packing Problem (BPP), which is known to be a NP-hard optimisation problem. In the literature there are several heuristics for the BPP [1]. Well known on-line heuristics are: next fit, first fit, best fit, etc. Off-line heuristics extend these approaches by applying initial sorting, resulting in: next fit decreasing, best fit decreasing, etc. Inspired by these main concepts, heuristics for the FPP have been developed. Despite FPP having a significant impact on system performance, there exist only a few works addressing the FPP. Most FPP algorithms mimic some BPP heuristic. [2] mimics *next fit decreasing*, where messages are sorted by their deadline. [3] mimics *best fit decreasing*, where messages are sorted by their periods, and the sorted message-list is processed alternately from the beginning and the end. In [4] messages are sorted by their offsets. [3], [4] combine the FPP with the scheduling problem. [5], [6] include the FPP into the TAP. Thereby FPP and TAP are formulated as a Mixed Integer Linear Problem (MILP).

Besides these differences, all state-of-the-art FPP algorithms share one common issue: The packing decision is made based on one condition only:

$$\text{message.size} \leq \text{frame.payload.left} \tag{1}$$

Sophisticated frame packing should be bandwidth demand efficient. This can be achieved by minimising the overhead data, thus the number of frames must be reduced. Therefore, as much data as possible should be packed into each frame. It seems the ideal situation is to have all frames fully utilised (i.e. the max. amount of payload data is used). The bandwidth consumption of a frame is:

$$bw_f = \frac{pay_f + oh_f}{T_f} \tag{2}$$

If several messages are packed into a frame, the frame must be sent at a rate that satisfies the rate of all packed-in messages. Thus, the message with the shortest period determines the frame's period.

$$T_f = \min_{i=m\in f} \{T_i\} \tag{3}$$

Consequently, the ideal situation is to have fully utilised frames and all messages inside a frame having the same (or very similar) periods. Since both the data size as well as the period of messages vary, the optimal situation represents a trade-off: If messages with different periods are packed into a frame, the frame must be sent at the shortest period, and thus some of the messages are sent more frequently than needed. This increases the bandwidth consumption. On the other hand, the more messages are packed into a frame, the less frames are needed. Thus less overhead data is sent. This reduces the bandwidth consumption. FPP heuristics from the literature, [2], [3], tackle this trade-off by trying to pack messages with *similar* periods. This is done by sorting the messages due to their period. However, packing decisions are not based on the optimality criteria, but only on the necessary condition (see equation 1). In order to improve frame packing, we derive optimality criteria for the packing decision. These address the trade-offs that have to be faced within the FPP. Based on these criteria, near-optimal frame packing configurations can be found.

A. *Optimality criteria*

Assume the following minimal example: There exists a frame that already has some messages packed-in. Another message needs to be packed-in and it can fit the existing frame. The question is: Should the message be packed into the existing frame (thus extending it), or should the message be packed into a new frame? The optimal decision can be taken by analysing the bandwidth demand of the two alternatives (left and right side of equation):

$$\underbrace{\frac{pay_f + s_m + oh_f}{T_f^{'}}}_{\text{extended frame}} = \underbrace{\frac{pay_f + oh_f}{T_f}}_{\text{existing frame}} + \underbrace{\frac{s_m + oh_f}{T_m}}_{\text{new frame}} \tag{4}$$

The fact the shortest message period determines the frame's period means adding a message may change the period of the extended frame $T_f^{'}$. Originally it is T_f.

$$T_f^{'} = \min \{T_f \cup T_m\} \tag{5}$$

Depending on the relation between T_m and T_f, there are 3 cases for this packing situation. For each of them, an optimal decision can be made.

Case I: $T_m = T_f$*:* If the periods of the frame and the message are equal, it is always beneficial to extend an existing frame as no additional overhead data is created.

$$\frac{pay_f + s_m + oh_f}{T} = \frac{pay_f + oh_f}{T} + \frac{s_m + oh_f}{T} \tag{6}$$

$$oh_f < 2.oh_f \tag{7}$$

Case II: $T_m > T_f \Rightarrow T_f^{'} = T_f$*:* The trade-off is: By extending the frame, the message will be sent more frequently than needed, but no additional overhead is created. By creating a new frame, additional overhead is created, but the message will not be sent too frequently.

$$\frac{pay_f + s_m + oh_f}{T_f} = \frac{pay_f + oh_f}{T_f} + \frac{s_m + oh_f}{T_m} \tag{8}$$

$$\frac{s_m}{T_f} = \frac{s_m + oh_f}{T_m} \tag{9}$$

At the threshold period of the message, the two alternatives perform equally.

$$T_m^* = T_f \frac{s_m + oh_f}{s_m} \tag{10}$$

Thus, the optimal decision is:

- $T_m < T_m^* \Rightarrow$ extending the frame is beneficial
- $T_m > T_m^* \Rightarrow$ creating a new frame is beneficial

Case III: $T_m < T_f \Rightarrow T_f^{'} = T_m$*:* The trade-off is: By extending the frame, the frame will need to be sent more frequently, but no additional overhead is created. By creating a new frame, the original frame will not be sent more frequently, but additional overhead is created.

$$\frac{pay_f + s_m + oh_f}{T_m} = \frac{pay_f + oh_f}{T_f} + \frac{s_m + oh_f}{T_m} \tag{11}$$

$$\frac{pay_f}{T_m} = \frac{pay_f + oh_f}{T_f} \tag{12}$$

The threshold period is:

$$T_m^* = T_f \frac{pay_f}{pay_f + oh_f} \tag{13}$$

Thus, the optimal decision is:

- $T_m < T_m^* \Rightarrow$ creating a new frame is beneficial
- $T_m > T_m^* \Rightarrow$ extending the frame is beneficial

B. *Improved frame packing heuristic*

The main issue of state-of-the-art frame packing heuristics is: During packing only the necessary packing condition (see equation 1) is checked. In case the periods of messages vary significantly, the approaches perform poorly, even if messages are sorted by their periods. To overcome this issue, two concepts have to be combined: First, the variation of message periods has to be minimised. This can be achieved by sorting the messages by their periods. Second, the packing decision must be taken by also incorporating the trade-off optimality criteria. This way, bandwidth demand can be reduced/minimised.

The proposed frame packing heuristic (see algorithm 1) addresses both aspects. Its structure is inspired by the *Fixed Frame Size* approach of [2] which mimics *next fit decreasing*. However, messages are not sorted by their deadline. Instead messages are sorted by their period, inspired by [3].

Within the *ExtendOrNew* method, the most beneficial decision is determined using the optimality criteria presented earlier. This way each packing step has minimal increase of bandwidth demand. Due to the NP-hard nature of the FPP, this approach cannot guarantee an optimal packing, however it finds packing configurations which outperform state-of-the-art approaches (as shown in section III).

III. Experimental results

The approach in section II makes no assumption about the type of bus, other than it is serial, including how it is scheduled and as such it should be generally applicable. However the Controller Area Network (CAN) protocol [7] is used here as it is widely used in embedded systems [2], [3], [7] and it was the protocol used in the existing FPP approaches [2], [3], [4].

Algorithm 1: Frame packing (based on optimal decisions)

```
Input: messages
sort(messages, T, increasing) /* sort by T [0..n] */;
frame = new frame;
foreach message do
    if $frame.payload.left \geq message.size$ then
        /* take most beneficial decision */;
        benefit = extendOrNew(message, frame);
        if benefit = extend then
            pack(message, frame);
        else if benefit = new then
            pack(message, new frame);
        end
    else
        pack(message, new frame);
    end
end
Output: frames
```

In order to gain statistically significant evaluation-results, 100 problem instances are evaluated for each point in the used specification-space. In total 20x 100 examples. The examples are synthesised in accordance to [2], [3], [4], [8] and in alignment with 2 real automotive examples (which unfortunately could not be made openly available).

- message size = 1 .. 16 bits (uniform)
- message period = 5 .. 1000 ms (uniform)
- bus baudrate = 125, 256, 500 kbaud/s
- nominal bus utilisation = 10 .. 20 %
- number of sending processing nodes = 1 .. 20

These synthetic problem instances are solved with the following approaches:

- simple (1 message = 1 frame)
- Sandstroem et al. [2]
- Poelzlbauer et al. (our approach)

These approaches treat the FPP as a separated problem and are targeted at minimising bandwidth demand, thus they are comparable. We are not comparing against [3], [4], [5], [6], since these approaches are combining the frame packing with the scheduling / task allocation problem and are targeted at maximising schedulability. Thus a comparison would result in misleading outputs. However, experiments will show that our approach increases schedulability, while minimising bandwidth demand. As the evaluation metrics we are using:

- bandwidth demand: Frame packing should minimise the bandwidth demand of the frame-set.
- frame payload: Frames should be highly utilised, in order to have a good data-to-overhead ratio. Thus the used frame payload should be close to maximum.
- message period variation: Messages should not be sent too frequently. Thus messages that are packed in the same frame should have similar periods. Message period variation inside a frame should be low.
- schedulability of frame-set: Frame packing should have a positive impact on schedulabilty. Fewer frames, preferably none, should miss their deadline.

The main reason for performing frame packing is to efficiently use the available bus bandwidth. Thus, the efficiency of frame packing algorithms can be measured by the resulting bandwidth demand of the frame-set.

$$bw = \sum_{f \in F} \frac{pay_f + oh_f}{T_f} \tag{14}$$

Figure 1 shows the resulting bandwidth demand for the 3 packing approaches. Each point represents the average over 100 problem instances. Statistical analysis using notched boxplots confirms that the differences are statistically significant due to a 95% confidence interval. Note:

- *application data* represents the load, due to messages only. This can be seen as a lower bound for the frame packing that can never be reached.
- *bus baudrate* marks the available bandwidth. This can be seen as an threshold for feasibility.

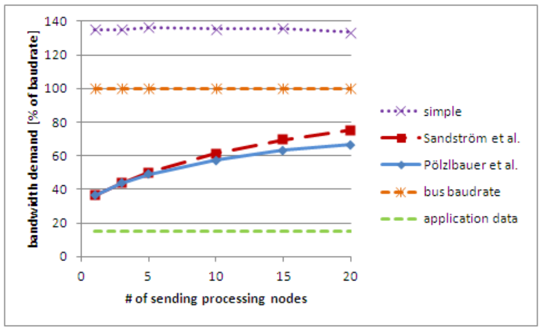

Fig. 1. Packing efficiency, due to resulting bandwidth demand. / s_m=1..16 bit; T_m=5..1000 ms; br=256 kbaud/s

The simple frame packing approach results in overloading the bus. Both, Sandstroem et al. and our approach, result in a significant lower bandwidth demand. However our approach outperforms Sandstroem et al's approach. General observations made include:

- For a low number of sending processing nodes, both approaches perform similar. For increasing number of sending processing nodes, our approach performs noticeably better.
- For lower bus baudrates, our approach performs significantly better. For higher bus baudrates, our approach still performs better, but the improvement becomes lower.

Table I shows the improvement of our approach over Sandstroem et al. in detail with respect to the number of nodes and bus bandwidth (on the left hand side of the table) and message sizes (on the right hand side of the table).

TABLE I
IMPROVEMENT OF POELZLBAUER ET AL. OVER SANDSTROEM ET AL.

	improvement [%]		
# nodes	125 k	256 k	500 k
1..3	0.0	0.0	0.0
5	5.9	2.3	0.0
10	14.4	6.2	3.3
15	13.8	15.0	2.4
20	17.6	16.2	6.2

message size [bit]	improvement [%]
1 .. 8	0 .. 18
1 .. 16	0 .. 18
1 .. 24	0 .. 19
1 .. 32	0 .. 16
1 .. 64	0 .. 6

Our approach performs well for a wide range of relevant message size (up to 32 bits). These ranges are the most relevant for the automotive domain, since most processed data is encoded within this range. For larger message sizes (32 to 64 bits) the improvement of our approach degrades. However, it can be noticed that for large message sizes (up to 64 bits) all packing approaches perform very similar, since almost only 1 message fits into 1 frame. It is important to note our approach is never worse than the existing FPP approaches, including simple packing.

The other important aspect of frame packing is the impact on the schedulability of the resulting frame-set. Since a frame consists of several messages, the frame must satisfy the deadlines of all packed-in messages. Thus the deadline of a frame must be set as:

$$D_f = \min_{i=m\in f} \{D_i\} \tag{15}$$

For priority assignment we use the deadline monotonic priority assignment policy and for calculating the WCRT of each frame we apply [7].

TABLE II
IMPACT OF FRAME PACKING ON SCHEDULABILITY / s_m=1..16 BITS; T_m=5..1000 MS; br=256 KBAUD/S; u=10 %; # SENDING PROCESSORS=10

	deadline missed	
	#	ratio
simple	230 of 651	0.409
Poelzlbauer et al.	0 of 93	0.0
Sandstroem et al.	0 of 91	0.0

Using the simple packing approach (1 message = 1 frame) results in a high number of frames. This means a high number of frames (especially low priority frames) miss their deadline (see table II). By using real packing approaches, the number of frames is significantly reduced, which again reduces the interference between frames. This results in a significant lower number of frames that miss their deadline.

Since both packing algorithms seem to perform equally in terms of schedulability, it is interesting to analyse the *robustness* of the frame-set. We define robustness as the ability to maintain schedulability, despite uncertainties in transmission time. Therefore we perform sensitivity analysis of the frame-set: We up-scale the transmission time (C) of frames, until the frame-set becomes unschedulable. The more we can up-scale C, the more robust is the frame-set.

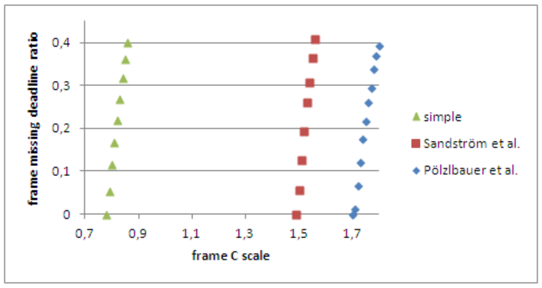

Fig. 2. Sensitivity analysis of frame-set with respect to frame transmission time

Figure 2 shows the ratio of missed deadlines with respect to the scaling of C. It demonstrates that our approach is much more robust (less sensitive). While the frame-set created by Sandstroem et al. can only be up-scaled by 1.49, the frame-set created by our approach can be up-scaled by 1.70. This represents an improved robustness of 14%. In addition, the unschedulability ratio of our approach increases less steep, which also indicates the higher robustness. At the top of the plot (ratio = 0.4) the utilisation of the frame-set exceeds 100%, thus the WCRT cannot be calculated any more. This happens at an up-scale factor of 1.57 and 1.81 accordingly. We can also down-scale C, in order to make the frame-set, which has been created by the simple packing approach, schedulable. We find that we need to down-scale C by 0.78.

IV. Conclusion

We have shown that using "real" frame packing algorithms significantly improves bandwidth demand and schedulability. Our improved approach outperforms the existing approaches with respect to all metrics. The approach works well for short message sizes between 1 and 16 bits (typically used today) as well as well longer ones (1 to 32 bits). It is also less sensitive to timing-variations. Future research may focus on additional optimisation objectives (e.g. min. end-to-end delay, min. jitter), emerging protocols (e.g. automotive Ethernet) and including the FPP into the TAP.

Acknowledgment

The authors would like to acknowledge the financial support of the "COMET K2 - Competence Centres for Excellent Technologies Programme" of the Austrian Federal Ministry for Transport, Innovation and Technology (BMVIT), the Austrian Federal Ministry of Economy, Family and Youth (BMWFJ), the Austrian Research Promotion Agency (FFG), the Province of Styria and the Styrian Business Promotion Agency (SFG). We also thank our supporting industrial (AVL List) and scientific (Graz University of Technology) project partners.

References

[1] E. G. Coffman, M. R. Garey, and D. S. Johnson, "Approximation algorithms for bin packing: a survey," in *Approximation algorithms for NP-hard problems*. Boston, MA, USA: PWS Publishing Co., 1996, pp. 46–93.
[2] K. Sandström, C. Norström, and M. Ahlmark, "Frame packing in real-time communication," in *International Conference on Real-Time Computing Systems and Applications (RTCSA)*, 2000, pp. 399–403.
[3] R. Saket and N. Navet, "Frame packing algorithms for automotive," *Embedded Computing*, pp. 93–102, 2006.
[4] P. Pop, P. Eles, and Z. Peng, "Schedulability-driven frame packing for multicluster distributed embedded systems," *ACM Transactions on Embedded Computing Systems*, vol. 4, pp. 112–140, 2005.
[5] W. Zheng, Q. Zhu, M. Di Natale, and A. Sangiovanni-Vincentelli, "Definition of task allocation and priority assignment in hard real-time distributed systems," in *IEEE International Real-Time Systems Symposium (RTSS)*, 2007, pp. 161–170.
[6] Q. Zhu, Y. Yang, E. Scholte, M. Di Natale, and A. Sangiovanni-Vincentelli, "Optimizing extensibility in hard real-time distributed systems," in *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2008, pp. 275–284.
[7] R. Davis, A. Burns, R. Bril, and J. Lukkien, "Controller area network (CAN) schedulability analysis: Refuted, revisited and revised," *Real-Time Systems*, vol. 35, pp. 239–272, 2007.
[8] K. Tindell and A. Burns, "Guaranteed message latencies for distributed safety-critical hard real-time control networks," Department of Computer Science, University of York, Tech. Rep., 1994, report no. YCS229.