# On Extensible Networks for Embedded Systems

Florian Pölzlbauer*, Iain Bate[†‡] and Eugen Brenner[§]
* Virtual Vehicle, Graz, Austria, florian.poelzlbauer@v2c2.at
[†] Department of Computer Science, University of York, York, United Kingdom, iain.bate@cs.york.ac.uk
[‡] School of Innovation, Design, and Engineering, Mälardalens University, Västerås, Sweden
[§] Institute for Technical Informatics, Graz University of Technology, Graz, Austria, brenner@tugraz.at

*Abstract*—**When designing a distributed computing-system, the communication networks are a key determining factor for system's performance. A common approach is to minimize bandwidth-consumption, while other important objectives – maintainability, extensibility, robustness – get less attention in the literature. In this work we provide a design-methodology how to efficiently balance these conflicting objectives. We build an initial network configuration by applying heuristics. Then, we refine this configuration by using optimization strategies which address the multi-objective optimization problem. By doing so, the network configuration not only satisfies the requirements of the current communication-demand, but it is also prepared to handle additional future communication-demand. Experimental results from an automotive case-study show that extensibility can be significantly improved (up to 44%) while trading only a little bandwidth-efficency (1% deteriation).**

*Keywords*-**real-time systems; frame packing; scheduling; automotive; extensibility; minimize bandwidth demand;**

## I. Introduction

Industrial embedded systems are usually in operation for a long life-span. During that time, it is not unusual that they are gradually upgraded in order to meet new requirements. Thus, *maintainability* and *extensibility* are key issues when initially designing these systems.

For distributed embedded systems, where applications are executed on different processing units and data is exchanged via communication networks, the configuration of these networks is a key determining factor for maintainability and extensibility. This is because the networks act as *global variables* for data exchange between the connected processing units.

Automotive systems are a typical example of distributed embedded real-time systems. Their networks use different arbitration protocols (e.g. LIN, CAN, FlexRay) and are themselves interconnected via gateways. For automotive systems, the requirements for maintainability and extensibility mainly stem from the evolutionary development approach. A "new" version of an automotive system is largely based on an "older" version which is extended / adapted towards the "new" requirements [1]. Thus, planing towards future modifications is crucial.

### A. Scope, Contribution & Outline

In this work we address the engineering task of *building a network configuration*. The challenge faced is the trade-off between *resource efficiency* and *extensibility* of the network configuration.

The contributions of this work to the community are:

- provide a set of change-scenarios which are most likely to occur during the system's life-span
- develop a set of strategies to address these scenarios when building the network configuration
- provide a methodology to balance the conflicting optimization objectives

The work is structured as follows: First, we describe the system model and define the network configuration engineering tasks. A motivation-example is used to highlight the associated challenges. Second, we provide change-scenarios which are likely to occur during the system's life-span, and derive strategies how these scenarios can be tackled. These strategies are then combined into an optimization framework which balances the conflicting objectives. Third, we present experimental results that show the efficiency of our approach. After reviewing related works, we finally draw our conclusions and identify future research aspects.

## II. Network Configuration

A set of processing units are interconnected via a communication network. Each processing unit executes a set of applications. An application reads data (either from sensors or the network), performs computations, and outputs data (either to actuators or the network). Data which is exchanged between applications is referred to as application-messages. These messages may either be exchanged processor-internally or via the network, depending on where the receiver-application is located.

### A. System Model

The *network configuration problem* is driven by the *network communication specification*: A set of application-messages which need to be sent via the network. A message is defined by its *data size*, the *period* at which data-values are generated, the *deadline* at which the message must arrive at the *receiver-processor*, and its *sender-processor*.

A message cannot be transmitted "raw" via the network. It has to be packed into a *network frame*. A frame consists of a header (which contains information needed for network arbitration and frame routing), the *payload* (i.e. the messages), and a tail (used for checking data integrity). Header and tail are referred to as *overhead*. According to the network protocol, the payload is of variable size, up to a defined *max. payload*.

Like messages, each frame also has period, deadline, sender- and receiver-processors.

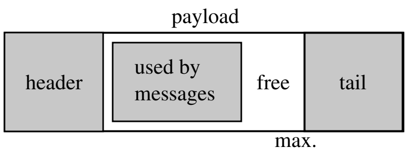


Fig. 1. Structure of Network Frame

In this work we assume that the data size of a message is smaller than the max. payload of a frame, thus several messages can be packed into one frame. This assumption is true for automotive systems (which are our main interest), but may also be true for other domains (such as rail, automation, aerospace, etc.) of embedded systems.

## B. Problem Statement

The engineering task of building a *network configuration* consists of two main steps: *frame packing* and *frame scheduling*. All decisions taken within these steps define a network configuration.

The *frame packing problem* (FPP) is defined as follows: A set of messages $M = \{m_1, m_2, ..., m_n\}$ which are output by a processor via a network interface, must be packed into a set of frames $F = \{f_1, f_2, ..., f_m\}$ subject to the constraints that the frames' max. capacity are not exceeded by the messages' data sizes, and the messages' deadlines are met.

Frame packing will define the frames' structure. Based on this, a schedule must be derived according to which the frames are transmitted. The *frame scheduling problem* (FSP) is defined as follows: A set of frames $F = \{f_1, f_2, ..., f_m\}$ transmitted on a network must be scheduled. For time-triggered networks (such as LIN or FlexRay), a set of time-slots $T = \{t_1, t_2, ..., t_u\}$ are defined, each time-slot is assigned to a processor, and the outgoing frames of each processor are assigned to the time-slots. For priority-based networks (such as CAN), each frame is assigned an unique priority-level $P = \{p_1, p_2, ..., p_m\}$. All these design decisions must be taken in a way so that the messages which are packed in the frames meet their deadlines.

## C. Design & Optimization Objectives

Since network bandwidth is a limited and valuable resource, frame packing is driven by the requirement to *minimize bandwidth consumption* of the frame-set [2], [3]. The bandwidth consumption of a frame-set is calculated according to equation (1). Both the frame's payload $pay_f$ and overhead $oh_f$ have to be transmitted at the frame's period $T_f$.

$$bw = \sum_{f \in F} \frac{pay_f + oh_f}{T_f} \tag{1}$$

The period has to be set to satisfy the periods of all packed-in messages, according to equation (2).

$$T_f = \min_{m \in f} \{T_m\} \tag{2}$$

For distributed systems, where communication is performed via networks, the delay that is introduced by the network has significant impact on the end-to-end delay of an application. Therefore, frame scheduling is often driven by the requirement to *minimize end-to-end delays* [4].

Tackling these issues at the same time leads to a conflicting multi-objective optimization problem. In order to minimize end-to-end delays, frames need to be sent at shortest possible periods. On the other hand, short periods increase the bandwidth demand.

The optimization problem becomes even more challenging if maintainability and extensibility needs to be addressed as well. Here, the network configuration not only has to satisfy the current communication specification, but it should also be capable of *incorporating future modifications and extensions*. Thus, we define extensibility as follows: The ability of a distributed real-time system to incorporate additional elements (i.e. messages and frames) later on, while maintaining the initial schedule. For time-triggered networks this implies that the initial time-slot assignment must not be modified. For priority-based networks it implies that the initial priorities must not be modified. In addition it implies that the initial frame packing is not modified.

## D. Example

Let us take a look at two small examples in order to highlight the trade-offs at hand. First, let us consider frame packing.

### Frame Packing

Assume a processing-unit that outputs 3 messages which need to be packed into frames before being transmitted via the network. We assume a max. payload of 8 byte, and 64 bit frame overhead (as applies to LIN and CAN).

- $m_1$: 8 bit, 50 ms
- $m_2$: 16 bit, 100 ms
- $m_3$: 32 bit, 150 ms

| **A: min. bandwidth** | | **B: extensibility** | |
|---|---|---|---|
| $f_1$: 8+16+32 bit, 50 ms | 2400 b/s | $f_1$: 8 bit, 50 ms | 1440 b/s |
| | | $f_2$: 16+32 bit, 100 ms | 1120 b/s |
| | 2400 b/s | | 2560 b/s |

At first it seems obvious that bandwidth-minimizing frame packing is the better approach. However, this changes if we take a look at what happens if a change occurs in the future. Assume an additional message needs to be packed.

- $m_4$: 16 bit, 100 ms

| **extending A** | | **extending B** | |
|---|---|---|---|
| $f_1$: 8+16+32 bit, 50ms | 2400 b/s | $f_1$: 8 bit, 50 ms | 1440 b/s |
| $f_2$: **16** bit, 100 ms | 800 b/s | $f_2$: 16+32+**16** bit, 100ms | 1280 b/s |
| | 3200 b/s | | 2720 b/s |

Due to the dense packing which is needed to achieve bandwidth minimization, additional messages do not fit the existing frame. Thus, a new frame needs to be created which causes additional overhead. However, if extensibility has been taken into consideration earlier, existing frames provide free space left, and additional messages can be added without needing to create new frames. Thus, no additional overhead is induced.

*Frame Scheduling*

Second, let us focus on frame scheduling. Assume 3 frames which need to be scheduled on a network. We assume a CAN network (which uses a priority-based arbitration schema). We further assume each frame's transmission time is 1ms.

- $f_1$: period=10 ms, deadline=4 ms
- $f_2$: period=10 ms, deadline=7ms
- $f_3$: period=10ms, deadline=9ms

It is well known that for a set of independent frames *deadline monotonic priority ordering* (DMPO) is the optimal scheduling approach. Thus, $f_1$ would get higher priority than $f_2$ and $f_3$. However, this only determines the priority ordering, but it does not state anything about absolute priority-levels.

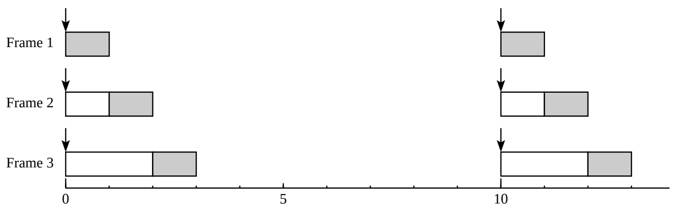


Fig. 2. 3 frames trying to arbitrate an idle CAN network simultaneously

Figure 2 shows how the 3 frames try to arbitrate the CAN network simultaneously. High priority frames (top) will win the arbitration process. Low priority frames (bottom) will be delayed accordingly.

| **A: dense priority-levels** | | **B: extensibility** | |
|---|---|---|---|
| $f_1$: priority=0 (highest) | 1ms | $f_1$: priority=0 (highest) | 1ms |
| $f_2$: priority=1 | 2ms | $f_2$: priority=5 | 2ms |
| $f_3$: priority=2 | 3ms | $f_3$: priority=10 | 3ms |

At first it seems irrelevant which absolute priority-levels are assigned to the frames. In both cases the deadlines are met. However, the impact of the choice becomes clear if an additional frame must be scheduled.

- $f_4$: period=10ms, deadline=6ms

| **extending A** | | **extending B** | |
|---|---|---|---|
| $f_1$: priority=0 (highest) | 1ms | $f_1$: priority=0 (highest) | 1ms |
| $f_4$: priority=**1** | 2ms | $f_4$: priority=**3** | 2ms |
| $f_2$: priority=**2** | 3ms | $f_2$: priority=5 | 3ms |
| $f_3$: priority=**3** | 4ms | $f_3$: priority=10 | 4ms |

According to DMPO the new frame $f_4$ needs to get a priority between $f_1$ and $f_2$. If there is no empty priority-level left between $f_1$ and $f_2$ (see case A) the priority-levels must be re-assigned. This may cause a significant number of changes which need to be propagated to all effected processors. In contrast, if the priority-levels are initially assigned in a way that there are empty priority-levels left (see case B) additional frames can be added without causing any priority-level re-assignments.

## III. Optimizing Network Configuration

The goal of this work is to address the delicate trade-off between *resource efficiency* and *extensibility* of a *network configuration*. Efficient resource usage is needed to satisfy the current communication specification, whereas extensibility tackles future changes thereof.

### A. Scenarios for Future Changes

Clearly, it is impossible to exactly predict which changes will occur in the future. However, by analyzing previous changes, reasonable estimations can be derived. Based on insight from our industrial partners, a set of likely change-scenarios are derived:

E-1: add a message
E-2: modify (increase) the data size of a message
E-3: modify (decrease) the period and/or deadline of a message
E-4: modify (add) receiver-processors of a message
E-5: add a frame
E-6: modify (decrease) the period and/or deadline of a frame
E-7: modify (add) receiver-processors of a frame

These scenarios are in alignment with modifications which may occur at the application level (e.g. adding software-components) and hardware level (e.g. adding processing-units). The change-scenarios can be grouped into 2 classes:

- modifications of messages: This will mainly have an impact on frame packing.
- modifications of frames: This will mainly have an impact on frame scheduling.

### B. Strategies for addressing Future Changes

Ideally, future modifications will cause few (or no) changes on the network configuration. Modifying existing messages and/or adding new messages will not result in the need to add additional frames. Modifying the timing-attributes of messages (and frames) will neither cause deadline-violations nor the need to re-build the schedule. Also, adding additional frames can be achieved with only minor changes to the schedule.

Based on the estimation of future change-scenarios, a set of strategies are derived which aim at achieving this ideal situation.

1) At the frame level *growth margins* must be planned. This means that frames are not fully packed with messages. Instead, a defined fraction of the frame's capacity is reserved for future demands. This reserve accounts for (a) adding additional messages, and (b) increasing message's data size.
2) At the schedule level *growth margins* must be planned as well. For priority-based networks (such as CAN) this means that not all priority-levels are used. For TDMA-based networks (such as LIN or FlexRay) this means that a set of non-used time-slots are added to the schedule, in

order to reserve resources for future demands. This reserve accounts for adding additional frames (which could be emitted by additional processing-units).

3) In order to account for modifications of timing-attributes (such as period and deadline) the schedule should be built in a way that it is *robust against timing-uncertainties*. This can be achieved by *minimizing sensitivity*.

In order to determine the extensibility of a network configuration (and thus, how close we can get towards the desired situation) a set of metrics are needed.

- *Frame growth potential* – The ability of adding additional messages or increasing the data size of messages can be used to determine the extensibility at the frame level. This metric measures this ability by analyzing the free space that is left in a frame.

$$gp_f = \frac{\text{payload free}_f}{\text{payload max}_f} \tag{3}$$

Since this gives a growth potential for each frame, an overall value is desirable. A reasonable estimation can be derived if the average growth potential is calculated.

$$gp_F = \text{avg}\,\{gp_f\}_{\forall f \in F} \tag{4}$$

If the minimum would be used, the entire configuration would be judged by a single frame only. This would be a overly pessimistic estimation.

- *Schedule growth potential* – The ability to add additional frames into a schedule has to be measured protocol-specific. For TDMA-based protocols, the number of non-assigned time slots can be used.

$$\text{gp}_{TDMA} = \frac{\|slots_{\text{free}}\|}{\|slots\|} \tag{5}$$

FlexRay-specific metrics which are tailored to the dynamic segment can be found in [5].

For priority-based protocols (like CAN) the number of additional frames which can be added before the system becomes un-schedulable can be used as a metric.

$$\text{gp}_{CAN} = \frac{\|F_{\text{added}}\|}{\|F\|} \tag{6}$$

Thereby, the frames' attributes (period, deadline, payload, priority) should be set randomly, but within specified ranges (such as: low, medium, high) derived from previous change-scenarios.

- *Robustness of schedule against timing-uncertainties* – Robustness analysis can be applied in order to ensure that schedulability is still guaranteed even if parameter-uncertainties exist. Uncertain transmission time (C) is linked to the frame's payload, and thus can be tackled by the *frame growth potential* (see above). Thus, the focus here is on uncertainties for periods (T) and deadlines (D). In order to determine the configuration's robustness, T and D of messages (and consequently also of frames) are decreased until the system becomes un-schedulable. This accounts for higher sampling rates and tighter deadlines. The more these parameter-values can be decrease the higher is the robustness.

$$r = \frac{\Delta D_f}{D_f} \quad \forall f \in F \tag{7}$$

## C. Balancing of Objectives

The goal of this work is to build a *network configuration* which trades-off the conflicting objectives of *resource efficiency* and *extensibility*. Extensibility can only be achieved if non-used resources (e.g. empty frame payload) are planned. However, this reduces resource efficiency. Both objectives have to be balanced accordingly. Thus, our optimization goal is as follows: Find a network configuration which satisfies all constraints, *minimizes the bandwidth consumption*, while it *maximizes the extensibility* and *maximizes the robustness against timing-uncertainties*.

A well-known approach to balance conflicting objectives is by combining them into a single objective-function. Here, the individual objectives are combined using a normalized weighted sum.

$$c = \frac{\sum_i c_i \cdot w_i}{\sum_i w_i} \tag{8}$$

Normalization is used so that each objective is within the same range (i.e. 0 to 1). Weights represent the importance to the individual objectives.

- $c_1$: bandwidth consumption of frame-set, according to equation (1)
- $c_2$: frame growth potential, according to equation (4)
- $c_3$: schedule growth potential, according to equation (5) and (6)
- $c_4$: robustness of schedule against timing-uncertainties, according to equation (7)

Weights for the objectives (see table I) were derived in accordance to discussions with our industrial partners. *Min. bandwidth consumption* and *max. frame growth potential* are equally balanced. This represents the trade-off between resource efficiency and extensibility. *Max. schedule growth potential* gets slightly less importance. This accounts for the following: Higher frame growth potential enables us to add additional messages without the need to generate additional frames for them, thus there is no impact on frame scheduling. Only if the existing frames can no longer incorporate additional messages, new frames must be generated. *Max. robustness* gets even less importance, since it is estimated that changes in period and/or deadlines are few.

TABLE I
BALANCING OF OPTIMIZATION OBJECTIVES

| **Objective** | **Weight** |
|---|---|
| min. bandwidth consumption | 10 |
| max. frame growth potential | 10 |
| max. schedule growth potential | 8 |
| max. robustness against timing-uncertainties | 5 |

### D. Optimization

During designing a distributed real-time system, building a network configuration is performed only one (or a few) time. It is not a task that is performed on a day-to-day basis. Thus, the time it takes to build the network configuration is not the key issue. One of our industrial partners stated: *"We don't care if it takes the entire night, as long as we get good results."*

Based on these boundary-conditions, we decide to apply solving methodologies which produce high-quality results within reasonable time. Thus we apply stochastic optimization. This class provides a set of search strategies, such as genetic algorithm, particle swarm optimization, and many more. Our choice is *simulated annealing* (SA), due to two reasons. First, it has proven well in related works [6] and second, it can easily be tailored to specific problems, since the user only has to set a few parameters (initial temperature, cooling factor, stop-criteria).

It is well known that the performance of SA depends on the *starting point* within the search space. In order to generate a reasonably good starting point, we apply initial solving heuristics. The initial network configuration is built as follows:

- bandwidth-minimizing frame packing [3]
- frames are scheduled according to DMPO

Bandwidth-minimizing frame packing according to [3] is used, since it represents the best-performing starting point (details will be presented in section IV-C). This initial network configuration is fed into the SA framework, within which it is refined. SA takes a network configuration, applies a modification to it, and evaluates if this modification improves the network configuration. This is repeated until the search converges towards an optimum. Evaluation is performed by applying the multi-objective objective-function, which was presented earlier. A modification between two network configurations is generated by applying one of the following *neighbour moves*:

- *Move a message into another frame* – Randomly choose a message, and then move it into another frame that originates from the same processor. If the original frame becomes empty by moving the message, the frame can be removed.
- *Move a message into a new/empty frame* – Randomly choose a message, and then move it into a new/empty frame with the same source-processor. The new frame is given a priority-level, close to the priority of the original frame. Either one above or one below.
- *Swap two messages between two frames* – Randomly choose a message. Randomly choose another message that originates from the same processor. Then, swap the packing of the two messages.
- *Swap the priority of two frames* – Randomly choose two frames. Then swap the priorities of the two frames.

These moves are performed according to the probabilities listed in table II. All neighbour moves which change the packing configuration are followed by an update-step where timing attributes (period, deadline, jitter) of the affected frames are refreshed.

After the search has converged towards an optimum, the *best obtained network configuration* is returned. In a final step, the frames' priority ordering is mapped to actual frame priority-levels (in case of CAN, these are determined by the IDs). In order to maximize the schedule's extensibility, the used priorities are evenly spread out across the available priority-levels (i.e. the ID-range) while maintaining their ordering. E.g. 50 priorities are spread out across 2048 IDs.

TABLE II
PROBABILITIES OF NEIGHBOUR MOVES

| **Neighbour move** | **Probability** |
|---|---|
| move message into other frame | 0.60 |
| move message into new/empty frame | 0.16 |
| swap two messages between two frames | 0.14 |
| swap priority of two frames | 0.10 |

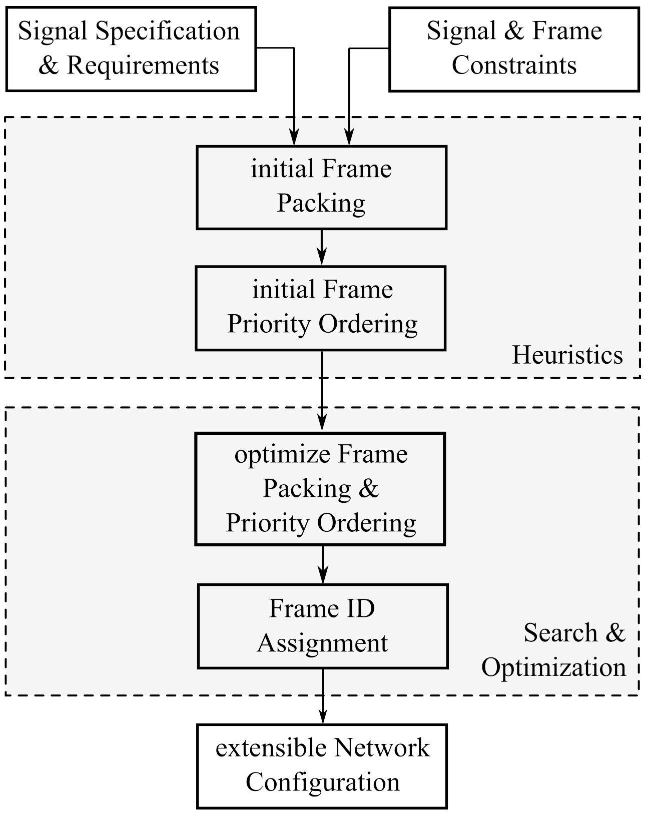

Signal Specification & Requirements
Signal & Frame Constraints
initial Frame Packing
initial Frame Priority Ordering
Heuristics
optimize Frame Packing & Priority Ordering
Frame ID Assignment
Search & Optimization
extensible Network Configuration

Fig. 3. Two-phased Solving Approach for Building a Network Configuration

Figure 3 depicts the entire solving approach. Note that it is demonstrated for priority-based networks (such as CAN). In case of TDMA-based networks, the scheduling-related aspects have to be adapted accordingly: Not priorities are modified, but time-slot assignments.

## IV. Experimental Evaluation

In order to evaluate the proposed methodology, it is applied to an industrial case study: an automotive in-vehicle network. Based on the communication specification, a network configuration is built. The results are analyzed and compared to state-of-the-art solving approaches.

## A. *Case Study: Vehicle CAN*

The network specification is taken from a *compact executive class (D-segment)* car. The *vehicle CAN* operates at 500 kb/s and uses CAN version 2a (which uses 11 bit frame IDs). 25 processing-units are connected to the CAN. Some of these processing-units represent *smart sensors/actuators*. The processing-units exchange 251 messages (which engineers would pack into 46 frames).

The communication specification can be described by the following statistics: Data size of messages is between 1 and 32 bits. 20% are boolean. 35% represent physical varialbes, encoded into 8, 12, 16, 24 or 32 bits. This is typical for embedded systems, where data encoding is in alignment with the accuracy of sensors and IOs. 55% are state/status variables, encoded into 2 to 7 bits. Figure 4 shows the histogram.

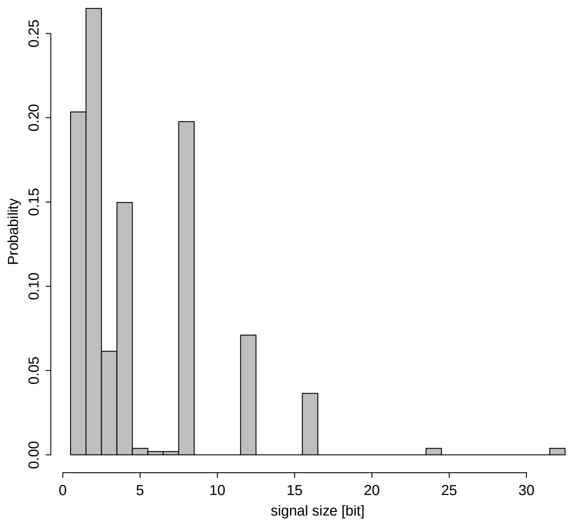


Fig. 4. Data Size of Messages

Messages are generated at rates between 10 and 400 ms. Table III shows that periods are not evenly distributed across this range, but only a set of dedicated periods are used. Messages' deadlines are equal to their periods.

TABLE III
PERIOD OF MESSAGES

| Period | Probability |
|---|---|
| 10 ms | 0.17 |
| 20 ms | 0.27 |
| 100 ms | 0.22 |
| 200 ms | 0.17 |
| 400 ms | 0.17 |

## B. *Results of Optimization Methodology*

As outlined earlier, an initial network configuration is built by applying heuristics. This initial configuration is then refined by applying the optimization strategies. While the initial configuration is generated within seconds, the optimization takes about 2.0 to 3.3 hours. During the optimization, about 30 000 configurations have been evaluated. About 95% of the time is consumed by the schedulability-tests [7]. A significant number of schedulability-test runs are needed for determining the *schedule growth potential* of a network configuration.

By comparing the initial configuration (which is tailored towards min. bandwidth consumption only) to the optimized configuration, we can determine by how much we can improve the network configuration. Table IV summarizes the results.

TABLE IV
IMPROVEMENTS ACHIEVED BY OPTIMIZATION

| Metric | Difference |
|---|---|
| min. bandwidth consumption | 1.0% deteriation |
| max. frame growth potential | 3.3% improvement |
| max. schedule growth potential | 44.4% improvement |
| max. robustness against timing-uncertainties | 0.7% deteriation |

As expected, the refinement towards *extensibility* comes at a certain cost in *resource efficiency*. The *bandwidth consumption* increases by 1%. At the same time, *robustness against timing-uncertainties* decreases by 0.7%. Balancing this, both *frame* and *schedule growth potential* increase. What we did not expect is the way in which they increase. According to the weights, we expected a bigger increase for frames than for the schedule. However, *schedule growth potential* increases significantly. This means that the schedule can incorporate a significant number of additional frames, but only few additional messages inside the existing frames.

One reason for the unexpected low increase of the frame growth potential may be rooted in the ratio between frame overhead and max. frame payload. For CAN both are 64 bit. Thus, not filling up frames causes a poor payload-to-data ratio, which again increases bandwidth demand.

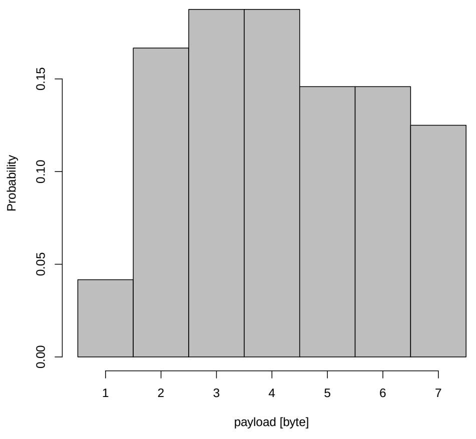
TABLE V
INITIAL VS OPTIMIZED NETWORK CONFIGURATION

| Metric | init. | opt. |
|---|---|---|
| bandwidth consumption [%] | 60.17 | 61.17 |
| frame growth potential | 0.703 | 0.679 |
| schedule growth potential | 0.552 | 0.307 |
| robustness against timing-uncertainties | 0.549 | 0.553 |

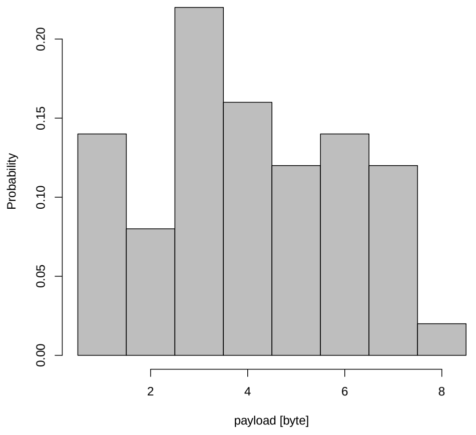
Table V provides deeper insight into the results. It compares the network configuration before and after optimization, broken down to the individual metrics. Note that all metrics are normalized between 0 and 1, whereas 0 represents the *best* and 1 represents the *worst*. For bandwidth consumption we scaled it to percent of the network's baudrate.

To provide a deeper insight in how frame growth potential is improved, the frames' payload can be analyzed. Figure 5 shows the frame payload histograms: according to the bandwidth-minimizing heuristic, and after refinement by the optimization strategies.

In the initial configuration frames are quite evenly filled. However, only few frames use 1 byte, and none used 8 bytes.



(a) bandwidth-minimizing packing by heuristic



(b) optimized packing by search framework

Fig. 5. Refinement of Packing – Impact on Frame Payload

After the optimization the frames fill-level have significantly changed. More frames only use a few bytes (1 to 3). The probability of using more bytes (4 to 7) decreases. This way, there is more free space left inside the frames. Consequently, these frames can incorporate additional messages in the future, thus they are more extensible. Interestingly – since unexpected – some frames even use 8 bytes.

## C. Impact of Initial Packing Heuristic

It is well known that the performance of SA is dependent on the starting position. This is why we investigate different heuristics for building the initial network configuration, and how this impacts on the *best obtained solution* found by SA.

Knowing that DMPO is already the optimal solution for scheduling independent frames, we focus on variations for *frame packing* only.

### *Bandwidth-minimizing Packing [2]*

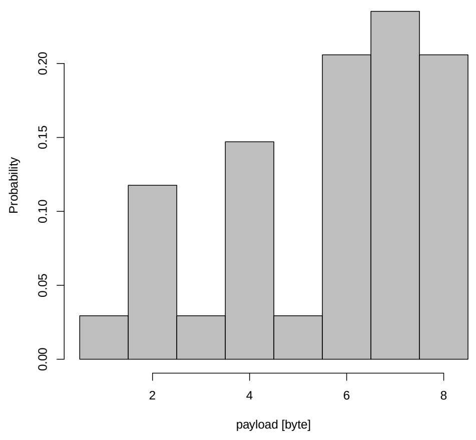
By applying bandwidth-minimizing frame packing according to [2] similar results than the ones discussed above can be achieved.

TABLE VI
INITIAL VS OPTIMIZED NETWORK CONFIGURATION

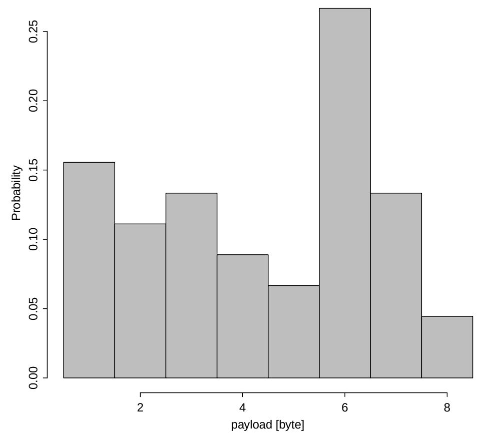
| **Metric** | **init.** | **opt.** | **Difference** |
|---|---|---|---|
| bandwidth consumption [%] | 66.25 | 67.21 | 0.95% deter. |
| frame growth potential | 0.953 | 0.750 | 21.3% impr. |
| schedule growth potential | 0.618 | 0.425 | 31.3% impr. |
| robustness against timing-uncertainties | 0.627 | 0.643 | 2.5% deter. |

Table VI shows the comparison between the initial and optimized network configuration. Again, slight deterioration in *bandwidth efficiency* and *robustness* are traded against significant improvements in *extensibility*. However, if we compare the absolute numbers to the previous results, we see that [3] outperforms [2].

Figure 6 shows the frame payload histograms. We see that in the initial packing 65% of the frames are densely packed (using 6 to 8 bytes), leaving only few free space for future growth. After the optimization, this is shifted towards less dense packing, and thus increased growth potential.



(a) bandwidth-minimizing packing by heuristic



(b) optimized packing by search framework

Fig. 6. Refinement of Packing – Impact on Frame Payload

*Simple Packing*

*Simple* frame packing is a widely used approach in the literature. It refers to putting each message into a separate frame. Clearly, this way the *frame growth potential* is maximized. However, this approach features a set of problems: The network is highly utilized (in our case it is even overloaded) and a high number of deadlines are missed.

TABLE VII
INITIAL VS OPTIMIZED NETWORK CONFIGURATION

| **Metric** | **init.** | **opt.** | **Difference** |
|---|---|---|---|
| bandwidth consumption [%] | 154.63 | 60.61 | 94% impr. |
| deadlines missed [%] | 73.6 | 3.9 | 69.7% impr. |
| frame growth potential | 0.141 | 0.648 | 361% deter. |
| schedule growth potential | - | - | - |
| robustness against timing-uncertainties | - | - | - |

Table VII shows the comparison between the initial and the optimized network configuration. Although the SA manages to improve bandwidth consumption, the final network configuration still misses some deadlines, and thus is infeasible. Since the system is not schedulable, *schedule growth potential* and *robustness* cannot be evaluated. Instead, we assumed the worst value (i.e. 1).

Figure 7 shows the frame payload histograms. Simple packing results in 80% of the frames only using 1 byte. After the optimization, this number drops to 40%. The remaining frames use 2 to 7 bytes.

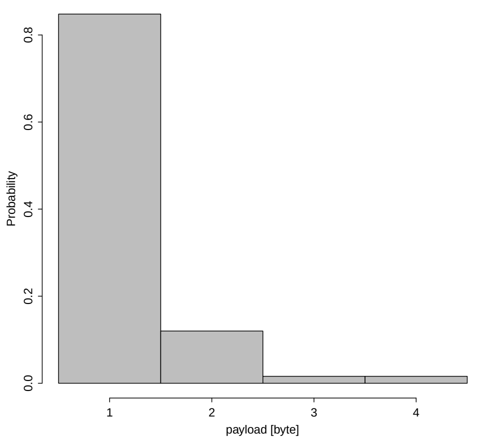
*Lessons learned*

Bandwidth-minimizing frame packing according to [3] combined with DMPO is the best performing heuristic to build an initial network configuration for a priority-based network. This configuration represent a reasonably good starting point for the refinement-phase, performed by the SA framework. Simple packing is a poor choice. It produces infeasible configurations, which even the optimization-phase cannot fix.

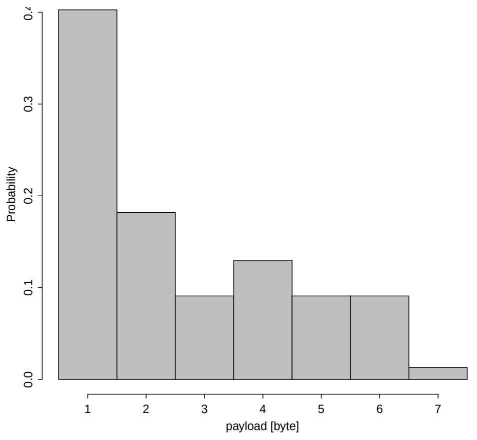
## V. Related Work

The research on *robustness* of real-time systems mainly focuses on the *sensitivity* of a system to uncertainties, changes and failures. The seminal work in this area is that of Punnekkat [8]. Punnekkat looked at by how much can individual tasks have their worst-case execution-time (WCET) increased before the system is no longer schedulable and which task is the most sensitive, i.e. which task's deadline is exceeded. He also looked at the number of re-executions, due to failures, that could be performed before the system was no longer schedulable.

Building on top of Punnekkat's approach, researchers used it as part of designing systems. In [9] sensitivity was used as an objective in a search algorithm such that robust task allocations are obtained. Later work by Natale [10] extended this work to look at how it could be done more efficiently using mixed integer linear programming (MILP). In [11] a measure of *robustness* is used that identifies the degree of error in key parameters which can be accommodated when performing task allocation. One parameter was by how much WCETs could be increased.



(a) simple packing by heuristic



(b) optimized packing by search framework

Fig. 7. Refinement of Packing – Impact on Frame Payload

The research on *extensibility* of real-time systems tackles the issue of *adding extra functionality* to systems (i.e. adding tasks as well as increasing the WCET of existing tasks). Emberson [12], [6] used a simulated annealing algorithm to design task allocations that were robust to change. Emberson's approach to develop a task allocation not only satisfied the baseline, or original, design problem but also satisfied as many potential change scenarios without any *unnecessary changes*. If an unnecessary change was needed then it was minimized. An necessary change would include adding a new task priority if a new task was introduced.

Researchers have adopted these approaches (which have originally been developed for task scheduling) to scheduling of communication networks. In [13] a time-triggered system is made robust to new frames (and tasks), and increases in the existing frame's worst-case transmission-times (WCTT) (as well as task's WCET). This work is probably the closest to ours. However, it does not address frame packing, which introduces additional challenges.

Research on *frame packing* focus on *minimizing bandwidth* *demand* [2], *maximizing schedulability* [14], or *maximizing reliability* [15]. To the best of our knowledge, no work addresses frame packing in the context of extensibility.

### *Open Issues in Related Work*

By combining frame packing with frame scheduling, a set of additional questions and challenges arise: Should extensibility be incorporated into frame packing, frame scheduling, or even both? How can we determine extensibility at the frame-level? Can frame packing handle timing-uncertainties?

In addition, extensibility and robustness are not the only objectives that need to be considered. Thus, additional questions arise: How to balance extensibility, robustness and resource-efficiency? By which step (i.e. frame packing, frame scheduling) can these objectives be tackled?

Our approach addresses these questions and issues by: We build an initial packing and schedule (based on state-of-the-art approaches), and then refine it towards balancing the conflicting objectives using optimization strategies.

## VI. Conclusion & Future Work

Communication networks are a key element for realizing distributed embedded systems. Building a configuration for these networks is a challenging engineering task. Several conflicting objectives have to be taken into account. On the one hand, resources (i.e. the network bandwidth) have to be efficiently utilized. On the other hand, the network configuration should be robust against uncertain parameters, as well as extensible towards future modifications.

In this work we presented an approach to address this trade-offs, and an algorithm to build a network configuration. Based on a set of potential change-scenarios, a set of strategies to address these scenarios were derived. These strategies were then incorporated into an optimization approach: An initial network configuration is build by applying heuristics. This configuration is then refined by applying our strategies (implemented in a simulated annealing optimization framework).

Experimental evaluation on an automotive case study shows that the proposed approach successfully addresses the trade-offs: Extensibility can be significantly improved (up to 44%), while almost no loss of resource efficiency and robustness occurs (about 1%).

In this work we treated each network as a stand-alone problem. In the future we want to evaluate, how we can apply our approach to multi-network problems, where several networks are interconnected via gateway-nodes. In addition, we want to evaluate how safety-related constraints [16] (e.g. two messages are not allowed to be packed into the same frame) impact on extensibility and robustness.

In this work we used a normalized weighted sum to balance the multiple objectives. It might be interesting to extend our approach, so that it not only returns a single "best" solution, but a set of pareto-optimal solutions. This would give the user more flexibility, and remove the need to set weights.

## 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 acknowledge the Swedish Foundation for Strategic Research (SSF) SYNOPSIS Project for supporting this work. We also thank our supporting industrial (AVL List) and scientific (Graz University of Technology) project partners.

## References

[1] D. Buttle, "Real-time in the prime time," in *Euromicro Conference on Real-Time Systems (ECRTS)*, 2012, keynote.

[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] F. Pölzlbauer, I. Bate, and E. Brenner, "Optimized frame packing for embedded systems," *IEEE Embedded Systems Letters*, vol. 4, no. 3, pp. 65–68, 2012.

[4] 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.

[5] R. Schneider, D. Goswami, S. Chakraborty, U. Bordoloi, P. Eles, and Z. Peng, "On the quantification of sustainability and extensibility of FlexRay schedules," in *Design Automation Conference (DAC)*, 2011, pp. 375–380.

[6] P. Emberson and I. Bate, "Stressing search with scenarios for flexible solutions to real-time task allocation problems," *IEEE Transaction on Software Engineering*, vol. 36, no. 5, pp. 704–718, 2010.

[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, no. 3, pp. 239–272, 2007.

[8] S. Punnekkat, R. Davis, and A. Burns, "Sensitivity analysis of real-time task sets," in *Advances in Computing Science – ASIAN'97*, ser. Lecture Notes in Computer Science, 1997, vol. 1345, pp. 72–82.

[9] I. Bate and N. C. Audsley, "Flexible design of complex high-integrity systems using trade offs," in *IEEE International Symposium on High-Assurance Systems Engineering (HASE)*, 2004, pp. 22–31.

[10] Q. Zhu, Y. Yang, M. Natale, E. Scholte, and A. Sangiovanni-Vincentelli, "Optimizing the software architecture for extensibility in hard real-time distributed systems," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 4, pp. 621–636, 2010.

[11] D. Gu, F. Drews, and L. Welch, "Robust task allocation for dynamic distributed real-time systems subject to multiple environmental parameters," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2005, pp. 675–684.

[12] I. Bate and P. Emberson, "Incorporating scenarios and heuristics to improve flexibility in real-time embedded systems," in *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2006, pp. 221–230.

[13] W. Zheng, J. Chong, C. Pinello, S. Kanajan, and A. Sangiovanni-Vincentelli, "Extensible and scalable time triggered scheduling," in *International Conference on Application of Concurrency to System Design (ACSD)*, 2005, pp. 132–141.

[14] R. Saket and N. Navet, "Frame packing algorithms for automotive applications," *Journal of Embedded Computing*, pp. 93–102, 2006.

[15] B. Tanasa, U. Bordoloi, P. Eles, and Z. Peng, "Reliability-aware frame packing for the static segment of flexray," in *International Conference on Embedded Software (EMSOFT)*, 2011, pp. 175–184.

[16] F. Pölzlbauer, I. Bate, and E. Brenner, "Efficient constraint handling during designing reliable automotive real-time systems," in *International Conference on Reliable Software Technologies (Ada-Europe)*, 2012, pp. 207–220.