DynOAA - Dynamic Offset Adaptation Algorithm for Improving Response Times of CAN Systems

Tobias Ziermann and Jürgen Teich
University of Erlangen-Nuremberg
Germany

Zoran Salcic
The University of Auckland
New Zealand

***Abstract*—CAN bus systems are used in many industrial control applications, particularly automotive. Due to growing system and functional requirements, the low capacity of the CAN bus and usually strict conditions under which it is used in real-time applications, applicability of CAN bus is severely limited. The paper presents an approach for achieving high utilization and breathes new life to CAN bus based systems by proposing a dynamic offset adaptation algorithm for scheduling messages and improving message response times without any changes to a standard CAN bus. This simple algorithm, which runs on all nodes of the system, results in excellent average response times at all loads and makes the approach particularly attractive for soft real-time systems. We demonstrate the performance improvement of the proposed approach by comparisons to other approaches and introduce a new performance measure in the form of a rating function.**

***Index Terms*—WCRT, Controller Area Network, CAN, response time, distributed embedded systems**

I. Introduction and Motivation

Many distributed embedded systems use a communication bus for communication between their components, usually called electronic control units (ECUs) or nodes. Examples of such distributed embedded systems are found in automotive and control applications. The communication bus has limited nominal capacity and bandwidth, and, in addition, the simultaneous accesses to the bus by multiple nodes result in conflicts that increase the response time of messages. In order to address this problem, a usual approach is to use the bus with a low load and utilization and, hence, reduce the likelihood of simultaneous accesses and conflicts.

Today, the most common bus for distributed real-time control systems is the controller area network (CAN) bus [2]. The CAN bus has not only the low bandwidth (maximum 1 Mb/s), but also potentially big latency, because messages with low priority get the access denied.

The primary motivation for our work is how to increase the utilization and at the same time reduce average response times of the CAN bus in automotive applicatision by adding more intelligence to the scheduling of messages. The proposed approach for message scheduling specifically targets soft real-time systems in which satisfying timing constraints is important, but missing the deadlines once in a while will not lead the overall system to malfunction. Current techniques for calculating the response times on priority-based serial buses are very pessimistic [3] and result in low bus utilization. The question we are trying to answer is: *How can the existing bus resources be better utilized by software modifications on the nodes while keeping the bus intact?*

A solution we propose is to distribute accesses from different nodes and application tasks to the bus in time and thus minimize the probability of simultaneous accesses and conflicts by taking into account the recent bus traffic history. The distribution in time is achieved by setting different offsets to message release times using software without changes in the message format and with no hardware modifications. The problem becomes more difficult due to the asynchronous nature of the distributed system and the lack of reference clock. Current approaches assign statically calculated fixed offsets to messages that are assumed to be synchronous by using off-line heuristics [6] that results in improved response times. However, they do not consider the dynamics of the system. In this paper, we propose a new method of dynamic adaptation of offsets based on monitoring of the network traffic, where each task that releases messages changes offsets on-line.

The rest of the paper is organized as follows. In Section II, we define the problem and place it into the context of related work. Section III gives the details of the dynamic offset adaptation algorithm and illustrates its operation on examples. Section IV compares the proposed approach with other approaches to message scheduling on CAN bus and introduces a new rating function as a measure for the quality of scheduling. Conclusions and future work are given in Section V.

II. Problem Definition and Related Work

In CAN [2], a message-oriented approach is chosen in the data link layer. Four different types of frames are used to transfer messages. We only consider the data frame, which is used for data exchange. Each data frame has its unique identifier. This identifier defines the message priority by which the bus access is granted. After sending the identifier, only the message with the highest priority is left and has exclusive bus access.

The system we target can be described by a set of nodes communicating over the bus as shown in Fig. 1. One or more tasks on each node periodically initiate the communication, i.e., release messages. The system as the whole is assumed to behave asynchronously, without a common time reference for the tasks on different nodes.

In our model, we abstract the tasks by considering only the mechanism used to release *messages* called a *stream*. A stream s_i can be characterized by a tuple (N_p, T_i, O_i) with $0 \leq O_i \leq T_i$, that is, by a node N_p the stream is running on, a period T_i (time between any two consecutive messages generated by stream s_i) and an offset O_i. The offset is relative to a global time reference. It can therefore drift

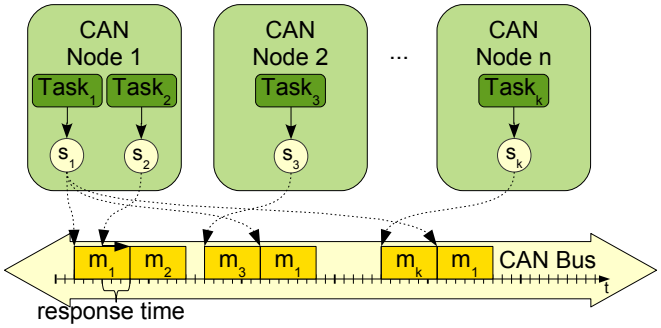

Figure 1. CAN-Bus based system model

over time, because the local time reference can differ from the global one. The *hyper-period* P is the least common multiple of all periods $lcm\{T_1, T_2, ..., T_k\}$. Assuming a synchronous system, the schedule is finally periodic with the hyper-period. A *scenario* consists of k streams. The priority by which access is granted to the bus is given by the numbering of the streams. A stream with a lower index, stream s_i, has higher priority than stream with a higher index s_{i+1}. We assume the priorities are set by the designer, typically according to the stream period so that a rate monotonic scheduling is achieved.

A *message* is a single release or CAN frame of the stream. The time between a message release and the start of its uninterrupted transfer over the bus is the *response time* of the message. In Figure 1 for example, the response time of message m_2 is three time slots, because it is delayed by the running message m_1. The *worst case response time (WCRT)* of a stream during a certain time interval is the largest response time of any message of the stream recorded during that time interval. If no time interval is specified, the WCRT is of the whole system lifetime, which is infinite, or of the simulation time in the case of simulation runs used to simulate system operation. In our model, we assume a discrete time with a minimal system time resolution defined in advance. All stream characteristic times are multiples of this minimal time resolution. Therefore, the messages can potentially occur at the same time, which results in larger response times. In our analysis, we assume an offset free system as defined in [4]. This means the offsets of individual streams are not bound by any constraints, but rather can freely be set by the designer. This approach potentially offers the avoidance of conflicts by setting the offsets appropriately in advance, before the system starts the operation such as in [5].

Offset-based approaches calculate offsets off-line prior to the system initialization. This is only valid if a common time reference is available and the traffic has repetitive behavior, the assumptions that do not hold for distributed systems. Algorithms to find the optimal static offsets are known to have exponential run-time [4], but near-optimal heuristics with reasonable run-time exist: Dissimilar Offset Assignment (DOA) [5]. In distributed systems, only the tasks and therefore also the streams on the same node are synchronous. In [6], the authors take advantage of this fact and propose an off-line offset assignment algorithm (OOA). The basic idea of the algorithm is to spread the release of messages by streams as far as possible each from the other on each node. According to the resulting schedule, the offsets are assigned to each stream, and WCRTs for the streams are reduced compared to the worst-case scenario, where all offsets are zero.

In our approach, we propose a dynamic adaptation of offsets, which change over time as the traffic on the CAN bus changes based on traffic monitoring carried out by individual nodes (streams). The adaptation of offsets is done by a simple algorithm, which does not require any significant computation, which we call the *dynamic offset adaptation algorithm* or simply DynOAA. The details of the algorithm as well as examples of its operation are presented in Section III.

III. Dynamic Offset Adaptation Algorithm (DynOAA)

In addition to the system model from Section II, we make some further assumptions. First, the DynOAA is targeted and analyzed for automotive scenarios, particularly those as described in [1]. Also, it does not deal with response times during system initialization.

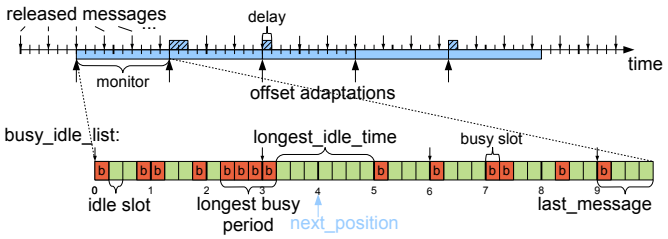

Figure 2. DynOAA illustration - timing diagram and busy_idle_list on a single node

Because our algorithm considers the entire distributed system with all message generating streams, the fact that one or more streams are on the same node can be abstracted and we will assume only one stream per node, which does not affect the generality of the presented results. The DynOAA is run on each node independently and periodically at certain instances of time that will be discussed in Section III-B. An illustration of the operation of the DynOAA for one stream is shown in Figure 2. In the upper part of the figure, on the top of the time line, the periodically released messages of the stream are indicated by small arrows. The larger arrows on the bottom of the time line indicate the instances when the adaptations start or when DynOAA is run, which includes a period of traffic monitoring. In Section III-A, we will first look into how the adaptation (new offset calculation) is done and in Section III-B, we will explain the adaptation triggering process.

A. Adaptation

Before the adaptation takes place, the bus is monitored by each stream for the time equal to the maximum period of all streams in order to have enough time to characterize the recent traffic on the bus. A list, from now on called *busy_idle_list*, is created. An example of it is shown in the lower part of Figure 2. It contains for each time slot during the monitoring an idle element if the bus is idle and a busy element if the bus is busy. The length of a time slot is in the first instance the transmission time of one bit. From the busy_idle_list, we can find the *longest_idle_time* and *longest_busy_time*, which are the maximum continuous intervals when the bus was idle or

busy, respectively. When finding these intervals, we consider the busy_idle_list as a circular list (by considering the first and the last time slot adjacent). The variable *last_message* denotes the amount of time passed between the current time and the time the last message was released for this particular stream. This value is needed to calculate when the next message would be released. The *next_position* is the time that indicates when in the next cycle a message should be scheduled. It is chosen in the middle of the longest_idle_time interval. The next message of the stream is then delayed, i.e. the offset is adjusted, so that a message is released at the time specified by next_position.

A pseudocode that describes the adaptation of the offsets is shown in Algorithm 1.

Algorithm 1 Offset adaptation

```
1:  monitor_time = 0
2:  while (monitor_time ≠ max_period - 1) do
3:     if (current_timeslot = busy) then
4:        busy_idle_list.add(busy)
5:     else
6:        busy_idle_list.add(idle)
7:     end if
8:     monitor_time = monitor_time + 1
9:  end while
10: if (first busy slot of LBP (see Section III-B)) then
11:    next_position = position in the middle of the
       longest_idle_time
12:    delay = (next_position + last_message) mod pe-
       riod
13: else
14:    delay = 0
15: end if
```

B. Time of Adaptation

In distributed systems, all streams are considered independent each from the other. If more than one stream starts to execute the adaptation simultaneously, there is a high probability that the value of the next_position at more than one stream will be identical. Instead of spreading, the message release times would in that case be clustered around the same time instance. Therefore, we need to ensure only one stream is adapting its offsets at the same time. Ensuring that only one node is adapting is achieved if all nodes make the decision whether to adapt or not based on a unique criterion based on the same information, the traffic on the bus in this case. The criterion we use is to select the stream belonging to the first busy slot of the *longest busy period* (LBP). The LBP is the longest interval of adjacent continuous busy slots without any idle slots. The idea is that this stream causes the biggest delay, because it potentially could have delayed all subsequent messages in the busy period and therefore should be moved first. If there are more than one LBP of equal length, the first one is chosen. If the monitoring phases of all nodes are synchronized, this mechanism ensures that all nodes choose the same stream.

IV. Results

Figure 3 shows WCRTs for all streams of a typical vehicle body CAN for different synchronicity assumptions. A description of scenario 1 is given in Section IV-A. Assuming that all streams are mutually asynchronous results in the highest WCRTs. Assuming that the streams on the same nodes are synchronous and their offsets are set by the DOA algorithm statically improves the WCRTs. The plots *synchronous* show the results for the WCRTs that are measured by a synchronous simulation. To simulate the asynchronous nature, 1000 different random offset values were run. This comparison shows a large overestimation gap when asynchronous assumption is applied. On the other hand, as the simulated results are obtained within a finite simulation time, there is no guarantee for the WCRTs to be upper bounds when obtained by simulation. However, we will demonstrate that DynOAA assigns the offsets so that it avoids reaching the WCRTs in typical automotive scenarios. Also, it most often finds optimal offsets, i.e. the response times of all streams are reduced to zero.

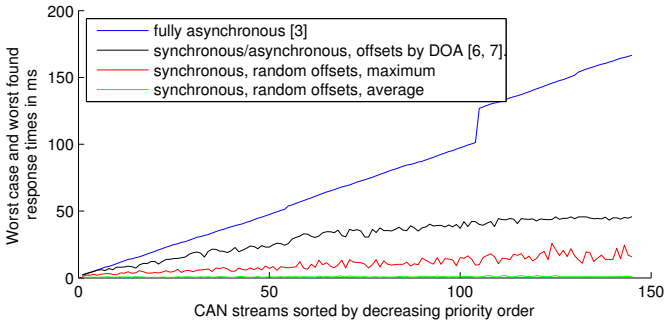

Figure 3. Worst case response times and worst observed response times for different synchronism approaches for scenario 1

A. Simulator

In order to evaluate the quality of our approach, we developed and used our own CAN bus simulator because the existing simulators are not capable of describing scenarios we used and extract the required properties. Our simulator is event-driven with the simulation step equal to one CAN bit. The full CAN protocol is reproduced by assuming worst-case bit-stuffing and the error-free case. We can only simulate the synchronous case, where all nodes have the same time base. This means if the offsets are fixed, the schedule repeats after time equal to the hyper-period. The asynchronous case is simulated by using different random initial offsets.

The scenarios we use for our experiments can be split into two groups. The first group consists of three close to real-world scenarios from the automotive domain, see Table I.

scenario	speed (kbits)	no of streams	mean workload	max period	source
1	125	145	0.51	2s	[8]
2	500	85	0.51	1s	[8]
3	500	56	0.48	2s	[7]

Table I
CLOSE TO REAL-WORLD SCENARIOS USED FOR THE EXPERIMENTS

The second group consists of synthetic scenarios generated by the Netcarbench [1] typical for the automotive domain with the bus load that can be freely adjusted. For this group of scenarios, we will always use a bus speed of 125 kbits with different average load generated by the streams.

B. Rating Function

In this paper, we propose the following rating function: $r = \sum_{i=1}^{k} \frac{WCRT_i}{T_i}$, where $WCRT_i$ is the worst case response time of the stream i or the worst response time found in simulations for the stream i, and T_i is the stream's i period. This function takes into account that streams with the large periods are more sensitive to large response times. As an example, the rating function for the motivating case in Figure 3 are shown in Table II.

scenario	r
fully asynchronous	26.52
synchronous/asynchrous	12.13
synchronous, maximum	4.35
synchronous, average	0.49
optimal	0

Table II
RATING FUNCTION VALUES FOR APPROACHES FROM FIGURE 3

C. Discussion

Figure 4 and 5 show the rating function over time for different scenarios. The experiments were always run with 10 different random offset initializations. The continuous lines in the plots represent the average of these 10 runs, while the vertical error bars indicate the maximum and minimum value of the rating function at that instance of time.

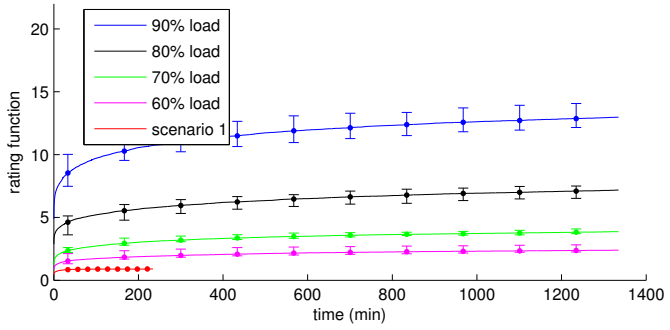

Figure 4. Rating function as a function of time for different application scenarios - maximal response times observed during the entire simulation

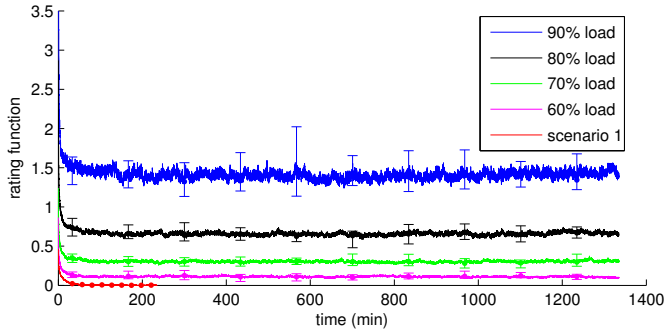

Figure 5. Rating function as a function of time for different application scenarios - response times of the last adaptation interval taken into account

Figure 4 shows the rating function that uses the maximal response time recorded since the start of the simulation. These values are comparable to the WCRTs of an analytical analysis [3], [6]. The rating function values for their analysis as shown in Table III are significantly worse. Even though we cannot prove that the response times have reached their real worst values, the experiments by simulations show that the rating function is increasing very slowly over time for a reasonable amount of simulated time (1400 min $\approx$ 1 day).

Scenario	1	2	3	60%	70%	80%	90%
Analytical [3]	26	13.5	7.3	41	55	94	130
DynOAA (entire sim)	1	1.5	1.2	3	4.5	7.5	14.5
DynOAA (last interval)	0	0	0	0.2	0.4	0.8	2

Table III
ANALYTICAL AND MAXIMUM DYNOAA RATING FUNCTION FOR DIFFERENT SCENARIOS

Figure 5 shows the rating function only for the last adaptation interval (2s in this case). We can see that it converges very fast to a stable value. The plot also shows that we are always improving compared to no adaptation case which is represented by the rating values at time zero. If we are dealing with a soft real-time system, where a few long response times can be tolerated, our method offers outstanding performance.

V. CONCLUSION AND FUTURE WORK

We propose a new approach to scheduling messages on a CAN bus that results in improved utilization of the bus and reduction of message average response times and as such particularly suited for soft real-time applications. The approach is based on dynamic adaptation of offsets used for message scheduling, which is done by each node (stream) independently. The assumptions of our modeling, particularly of asynchronous nature of nodes/streams on the CAN bus, are more realistic than those used in previous approaches. The proposed approach results in much better performance under all bus loads, shown by simulation. Those reduced, and sometimes optimal, response times are achieved without the need for specific prior knowledge. The approach also offers flexibility of network configuration change as it adapts message offsets on-line, based on the observed traffic on the bus. Our future work includes analysis of selective offset adaptation on a subset of nodes, taking into account system initialization as a part of the fully automated scheduling approach, exploring other rating functions for comparison and measurement of quality of the scheduling and the extension of the approach to the other types of networks beyond CAN.

ACKNOWLEDGMENT

Supported in part by the German Research Foundation (DFG) under contract TE 163/15-1.

REFERENCES

[1] C. Braun, L. Havet, and N. Navet. NETCARBENCH: A benchmark for techniques and tools used in the design of automotive communication systems. In *7th IFAC International Conference on Fieldbuses and Networks in Industrial and Embedded Systems*. Citeseer, 2007.
[2] CAN. Controller Area Network. http://www.can.bosch.com/.
[3] R.I. Davis, A. Burns, R.J. Bril, and J.J. Lukkien. Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised. *Real-Time Systems*, 35(3):239–272, 2007.
[4] J. Goossens. Scheduling of offset free systems. *Real-Time Systems*, 24(2):239–258, 2003.
[5] M. Grenier, J. Goossens, N. Navet, et al. Near-optimal fixed priority preemptive scheduling of offset free systems. In *14th International Conference on Real-time and Network Systems*. Citeseer, 2006.
[6] M. Grenier, L. Havet, and N. Navet. Pushing the limits of CAN-scheduling frames with offsets provides a major performance boost. In *Proc. of the 4th European Congress Embedded Real Time Software (ERTS 2008), Toulouse, France*. Citeseer, 2008.
[7] T. Herpel, K.S. Hielscher, U. Klehmet, and R. German. Stochastic and deterministic performance evaluation of automotive CAN communication. *Computer Networks*, 53(8):1171–1185, 2009.
[8] RTaW-Sim. Real-time at Work CAN Simulator. http://www.realtimeatwork.com/.