Optimized Resource Dimensioning in an embedded CAN-CAN Gateway

Jörg Sommer and Rainer Blind[1]

University of Stuttgart, Institute of Communication Networks and Computer Engineering
Pfaffenwaldring 47, 70569 Stuttgart, Germany
Email: joerg.sommer@ikr.uni-stuttgart.de

***Abstract*— The Controller Area Network (CAN) is a robust, low-cost, and simple event-triggered technology for connecting electronic control units in the manufacturing industry and vehicles. Today's real-time control systems are distributed over a multitude of CAN systems (domains) which are connected via embedded gateways. A failure or overload situation in the gateway can affect several domains. Furthermore, gateways often become bottlenecks between the domains and in the case of CAN buses they can pose further problems with respect to the priority-based network access method. Due to this access method and the limited resources (e.g., buffer capacity) embedded CAN-CAN gateways have to be dimensioned accurately. Otherwise, unacceptable processing delay and message loss within the gateway can occur.**

The main contribution of this paper is to investigate the optimized dimensioning of an embedded CAN-CAN gateway with regard to minimizing gateway resources in terms of processing and buffer capacity and decreasing message loss at the same time. For that purpose a CAN bus and a gateway model are described and used to investigate scenarios with two domains connected via a gateway.

I. Introduction

In February 1986, Robert Bosch GmbH introduced the serial bus system *Controller Area Network* (CAN) to reduce the size of the wiring harness. CAN improved the weight, reliability, serviceability, and installation. Due to its robustness, its low cost, and the simple-event triggered communication mechanism CAN is one of the dominating bus protocols used for real-time control systems, not only inside vehicles, also in machine control and factory automation, etc.

Real-time control systems are often distributed over a multitude of CAN systems (domains) which are connected via gateways. The reasons for such multi-domain systems are manifold and depend on the field of application. One of the main targets is the reduction of complexity. Furthermore, in the case that one bus fails, other buses are not affected. In-vehicle domains with different emphasis on safety and reliability need to be kept separate. In manufacturing areas the network is partitioned into small parts, since a shorter bus length enables a higher transfer rate due to the carrier sense multiple access scheme of CAN. Finally, a domain can group technological, topological, and functional *electronic control units* (ECUs).

[1]At the time of writing, Rainer Blind was a student at the IKR. Now, he is with the Institute for Systems Theory and Automatic Control, University of Stuttgart, Stuttgart, Germany, as a scientific staff member.
Email: rainer.blind@ist.uni-stuttgart.de

The independent domains are interconnected by an embedded CAN-CAN gateway. In embedded system design engineers have to optimize the resources, reducing the size and costs of the components. Keeping the costs low means reducing the needed resources to a minimum and designing it as simple as possible. CAN systems are mass-products, so the cost savings can be multiplied by millions of items. On the other hand, a failure or overload situation of a gateway can affect one or more domains. To sum up, fulfilling these requirements leads to a trade-off between deployed resources (e.g., CPU resources or buffer capacity) and costs. Managing this trade-off is one of the challenges in designing embedded systems and in our case a CAN-CAN gateway.

For example, shortened CPU resources can increase the forwarding time of a message, whereas in real-time control systems a predictable end-to-end message delay is required. Additional delay can be introduced by the gateway due to the priority-based network access method of CAN buses. Longer delays or message losses can destabilize a real-time control system [14]. Thus, in this paper possibilities to decrease the delay in the gateway are investigated. Furthermore, the impact on message delay caused by additional traffic will be considered.

Beside the CPU resources the buffer capacity is a further important, limited resource. Today, an embedded CAN-CAN gateway is usually integrated into a single-chip microcontroller with a fixed buffer size. In many cases, this buffer capacity has to be allocated individually to the input and output queue (e.g., [6]) depending on the connected domains. If the connected buses have different transfer rates, the gateway has to provide enough buffer capacity in the output queue to the low-speed bus, otherwise message loss can occur. However, even in the opposite direction some buffer capacity is necessary to store low priority messages until they win the arbitration. This asymmetrical assignment of the limited buffer capacity to avoid message loss is an issue and will be addressed in this paper.

Among other things, the buffer capacity can impact the loss probability and the mean waiting time of messages. In real-time control systems, it is preferred to drop a message instead of queuing and wasting resources while its information gets obsolete. Therefore, this trade-off between loss probability and mean waiting time will be investigated, too.

Motivated by the priority-based network access method and the aforementioned trade-offs the objective is the optimized resource dimensioning of a CAN-CAN gateway with emphasis on processing and buffer capacity and decreasing message loss at the same time. For that purpose, first an overview of related work is given. In section three and four a functional description of a CAN system and a CAN-CAN gateway are given. In section five models are described and used to investigate scenarios with two domains connected via a gateway. Then, section six shows the results that have been achieved with a simulation representing a general interconnection scheme in its simplest form by using two buses with different data rates and one gateway. In this section estimations about how an embedded CAN-CAN gateway should be dimensioned with respect to processing time and buffer capacity are given. Section seven shows how the gateway must be dimensioned to handle a burst of high priority messages. Finally, a conclusion and an outlook is given in section eight.

II. Related Work

The diversity of network interconnection leads to many publications. In [1], the author lists about 380 papers, documents, and books dealing with the various aspects of network interconnection.

Only a few papers discuss the interconnection of CAN-CAN systems. In [2], the authors address the design and implementation of a bridge which provides a selective frame retransmission function in interconnected CAN systems. In [4], the author describes general aspects and strategies for data transfer between CAN network subsystems, but various aspects of dimensioning the resources in a CAN-CAN gateway are not addressed.

In [3], the authors analyse the performance of bridged CAN systems using the benchmark of the Society of Automotive Engineers similar to our work. Additionally, in this paper, two CAN systems with different transfer rates are connected and the message loss probability when reducing the buffer capacity is investigated, too.

III. Controller Area Network

The *Controller Area Network* (CAN) is the most widely used automotive network for communication between *electronic control units* (ECUs). It was developed in the 1980s by Robert Bosch GmbH, a German automotive supplier, as a communication bus for in-vehicle electronics. Although, designed for vehicle applications, it was adopted to different scenarios and is nowadays used for embedded networking, machine control, factory and building automation, medical electronics, etc. Bosch's original specification [10] was submitted for international standardization in 1991 and later became an ISO standard documented in [8].

CAN is a robust, low-cost, and simple event-triggered bus system. It uses *Carrier Sense Multiple Access with Collision Resolution* (CSMA/CR) as medium access control mechanism. Simultaneous access of several nodes to the bus is detected and a collision is resolved in such a way that the message with the highest priority wins the arbitration process unmodified and the others terminate the transmission. This means that the transmission of the message continues after the arbitration and is not restarted — no further delay due to collision resolution is introduced. CAN is non-preemptive, i.e., a message that won the arbitration process will always be transmitted completely without regard to higher priority messages that arrive at a later point. Due to its priority scheme and the partially stochastic arrival process of higher priority messages, CAN can not guarantee deterministic response times for messages with lower priorities. Worst case response times can only be guaranteed for deterministic and other special arrival processes.

A CAN message frame consists of an arbitration field, control field, data field, CRC field, ACK field, and an end-of-frame field. The priority of a CAN message is determined by the identifier (a part of the arbitration field) which has a size of 11 bit (Standard CAN) or 29 bit (Extended CAN) respectively. Identifiers corresponding to low binary numbers enjoy a high priority and vice versa. Each identifier relates to a message with a certain content. Therefore, no identifier can occur more than once on a single bus. The identifier is also used by all receiving nodes to detect whether the message is relevant for them since CAN does not use addressing and all messages are broadcasted.

CAN uses a bit-stuffing mechanism for synchronization and error detection which increases the frame size by up to 20% and thus reduces the throughput. After five consecutive equal bits the sender inserts a stuff bit into the bit stream. This stuff bit has a complementary value which is removed by the receivers [8].

The CAN specification limits the maximum transfer rate to 1 MBit/s. Typical transfer rates in manufacturing and automotive environments are 500 KBit/s, 250 KBit/s, and 125 KBit/s. More detailed descriptions of CAN can be found in [5] and [15].

IV. CAN-CAN Gateway

The complete communication system consists of a number of ECUs which are connected to different buses. These compounds build the domains and are interconnected by gateways. What requirements has to fulfill a gateway in a manufacturing environment or an embedded in-vehicle CAN system to transfer cross-domain data between the domains?

The CAN-CAN gateway interconnects the domains on the *Logical Link Control* or the *Medium Access Control* level in the protocol hierarchy. Normally, an interconnection unit working on this protocol level is called a bridge. However, in a CAN-CAN gateway certain messages with less than 8 bytes payload might be combined/packed to a new message. Thus, this interconnection unit implements functionalities on application level.

Gateways connect two or more buses and pass data from one bus to the others. Passing data includes simple message forwarding as well as assembling new messages from the data of received messages before forwarding. Gateways introduce additional delay due to message processing: The destination domain has to be identified, the message itself has to be processed and finally access to the bus must be won. If an incoming message must be forwarded to several domains copying is necessary. If the message should be available on several domains, it can not be simple copied. One reason is that the message identifiers must be adapted to each domain [4]. In this case the message identifier must be changed within the gateway. This mapping is stored in a look-up table.

As mentioned before, if the connected buses have different transfer rates the gateway must provide enough buffer capacity in the direction from the fast to the slow bus, otherwise message loss can occur.

To sum up, gateways are a necessity producing additional costs and weight while not directly providing a customer-observable benefit. This requires implementing embedded gateways whose performance in terms of load, delay, and message loss have to be sensitively assessed. On the one side, the gateway resources should not be oversized due to the significance of the costs in the automotive industry. On the other side, the throughput and robustness must be guaranteed even with an increasing workload. Hence, it is extremely important to optimize their performance while minimizing their disadvantages.

V. Model

A. CAN model

In this paper the model presented in [11], [12], and [13] is used. It uses a single non-preemptive server as a model of the bus. The transfer rate on the bus is determined by the server holding time $T_{\mathrm{H,CAN}}$ where the holding time equals the transmission time of a CAN message. The bit-stuffing mechanism can be taken into account by using a higher $T_{\mathrm{H,CAN}}$ or another suitable distribution function. A priority multiplexer simulates the arbitration of the CAN by always selecting the queue with the lowest message identifier at its head first due to the fact that identifiers with low numbers enjoy a high priority.

B. Gateway Model

With respect to CAN buses, two types of gateways exist: (1) gateways which connect two or more CAN buses, and (2) gateways which connect CAN buses to buses with different technologies, e.g., Ethernet or FlexRay. Although, we only consider the first case in this work, our model shown in Figure 1 is universal and considers both cases. In the case of heterogeneous connections the gateway's task is more elaborate: It has to provide different media access mechanisms, efficient mechanisms for address translation or protocol adaptation, and it must possibly adapt to QoS requirements.

The gateway model shown in Figure 1 is based on the *Store-Modify-Forward-Principle*. Messages are received from the source bus, possibly modified by the server and then sent to the destination bus. Thus, the gateway can be decomposed into three major elements: (1) *receiving queue (Rx-buffer)*, (2) *server with phase duration* $T_{\mathrm{H,GW}}$, and (3) *transmission queue (Tx-buffer)*.

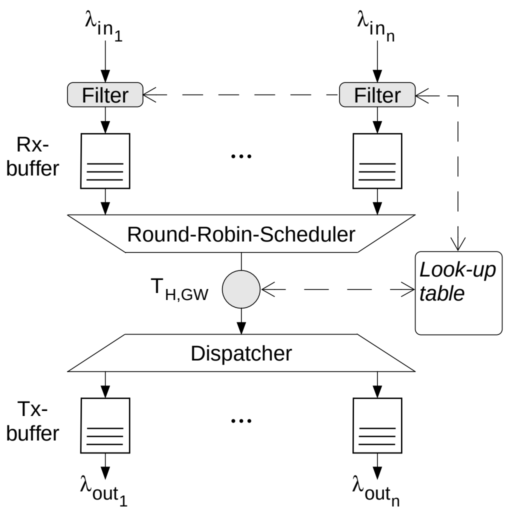

λ_{in_1}
λ_{in_n}
Filter
Filter
Rx-buffer
...
Round-Robin-Scheduler
$T_{\mathrm{H,GW}}$
Look-up table
Dispatcher
Tx-buffer
...
λ_{out_1}
λ_{out_n}

Fig. 1. Gateway model with filter, receiving queue, server for identifier translation, and transmission queue.

The parameter n describes the number of attached buses. The filter verifies with the help of the *look-up table* whether an incoming message has to be forwarded to another bus or if the gateway itself is the destination. However, the filter's processing time $T_{\mathrm{H,Filter}}$ has to be much smaller than the transmission time of the respective bus $T_{\mathrm{H,CAN}}$ to avoid message loss caused by blocking. This filter functionality is implemented in hardware and done *on-the-fly* while receiving the message. Thus, we assume $T_{\mathrm{H,Filter}} \rightarrow 0$ and the filter is only considered as a functional unit without any impact on the gateway's performance and its message loss.

The *Round Robin Scheduler* guarantees a fair balanced scheduling for all connected buses. Messages can simply be passed through by the gateway, or the identifier can be altered by means of the look-up table (this corresponds to message recomposition).

The dispatcher routes the message to the corresponding destination queue. If the message is destined to more than one outgoing bus, it will be copied and enqueued correspondingly.

The delay of an individual message in the gateway can be calculated as

$$T_{\mathrm{GW}} = T_{\mathrm{Rx}} + T_{\mathrm{H,GW}} + T_{\mathrm{Tx}}, \qquad (1)$$

where T_{Rx} is the delay an incoming message has to wait until the message is served. $T_{\mathrm{H,GW}}$ is the gateway holding time and T_{Tx} is the delay until an outgoing message can be sent in the destination domain.

VI. Resource Dimensioning

By means of simulation we investigate how an embedded CAN-CAN gateway must be dimensioned with respect to its processing time and buffer capacity. The model of section V was implemented and simulated with the *IKR Simulation Library* (IKR SimLib) [7]. The IKR SimLib is an object-oriented class library for event-driven simulation.

The integration of additional traffic into an existing CAN system is investigated. The existing system consists of two CAN domains with different transfer rates. We assume CAN messages with a constant payload size of 8 bytes and a bit-stuffed message size of 125 bit including header and tail. In the high-speed domain (500 KBit/s) the bus holding time is constant with $T_{H,CAN_{500}} = 0.25ms$ and in the low-speed domain (125 KBit/s) with $T_{H,CAN_{125}} = 1ms$.

In order to show principle effects of the system we limited ourselves to a simple traffic characterization. Both domains contain three priority classes with a Markovian arrival process. The highest and lowest priority generate 10% bus utilization, the third source 40% and simulates all other background-traffic. Thus, each domain has a bus utilization of $\rho = 60\%$. In general the bus utilization ρ can be calculated as

$$\rho = \sum_{i=1}^{n} \lambda_i \cdot T_{\mathrm{H,CAN}}, \tag{2}$$

where n is the number of priority classes and λ_i is the arrival rate of messages with priority i.

The aforementioned additional traffic has to send messages from the high-speed to the low-speed domain. This traffic is modeled as a Markovian arrival process with a mean interarrival time of 25 ms which implicates an increased traffic of 1% in the high-speed and 4% in the low-speed domain.

A. Impact of the Identifier

As mentioned before, in a CAN system the low priority messages have to wait until all high priority messages are handled. For integrating a novel application a new message identifier (priority class) has to be assigned. There is some degree of freedom in choosing this priority. If a high priority is chosen then the messages will get a fast bus access, but all other messages have to wait longer. Using a lower priority the messages will get a longer waiting time. In each domain the mean waiting time $E[T_{\mathrm{W,i}}]$ of a message with priority i (P_i) can be calculated as follows [9]:

$$E[T_{W,i}] = \frac{\rho}{2(1-\varrho_{i-1})(1-\varrho_i)} \cdot T_{\mathrm{H,CAN}}, \tag{3}$$

with

$$\varrho_i = \sum_{j=1}^{j=i} \rho_j \qquad \text{and} \qquad \rho_j = \lambda_j \cdot T_{\mathrm{H,CAN}}, \tag{4}$$

where ρ_j is the bus utilization generated by P_j messages.

The mean waiting time $E[T_{\mathrm{W,i}}]$ depends on the bus utilization generated by all higher priority messages (ϱ_{i-1}), its own messages (ϱ_i), and also on the overall bus utilization ρ. Using a low priority for the novel application increases the bus load ρ and thus the mean waiting time of all messages marginally. In the case that the highest priority is assigned to the novel application the mean waiting time for all other messages is significantly increased due to the priority-based bus access.

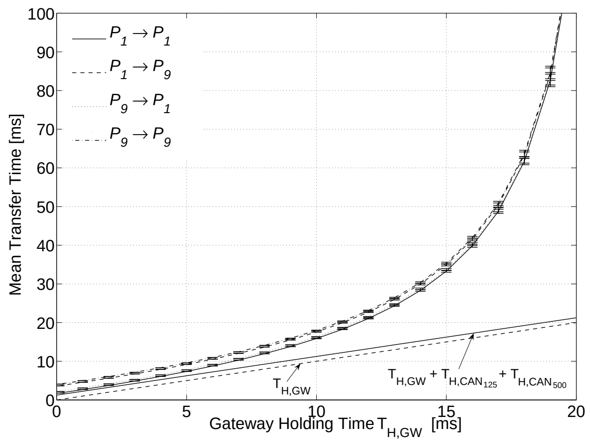


Fig. 2. The transfer time from the start to the destination domain depending on the gateway holding time. Without any queuing effects, the transfer time would be $T_{\mathrm{H,GW}} + T_{\mathrm{H,CAN}_{125}} + T_{\mathrm{H,CAN}_{500}}$.

B. Gateway Holding Time

Since the transfer time depends on the gateway holding time, now this dependency is investigated. Figure 2 shows the mean transfer time from the start to the destination domain depending on the gateway holding time. On the x-axis the gateway holding time $T_{\mathrm{H,GW}}$ and on the y-axis the mean transfer time is shown. $P_i \rightarrow P_j$ means that the messages of the additional traffic have the priority i in the source and the priority j in the destination domain; 1 is the highest and 9 the lowest priority in the entire system.

As expected the mean transfer time increases with the gateway holding time $T_{\mathrm{H,GW}}$. Except for the case of a minimal utilized gateway its holding time and the resulting queuing effects dominate the mean transfer time. As mentioned before, the existing system should be disturbed as less as possible. Since we investigate the dimensioning of the gateway resources, and not the optimal priority assignment, we will consider low priority messages for the additional traffic.

Using a powerful gateway leads to a lower transfer time, since no waiting time in the input queues occurs. Increasing the gateway holding time $T_{\mathrm{H,GW}}$ results in longer input queues and transfer time. As can be seen in Figure 2 this queuing effect dominates the mean transfer time if $T_{\mathrm{H,GW}} > 15ms$. Due to the mean interarrival time of $1/\lambda = 25ms$ for the additional traffic the gateway utilization is $\rho_{\mathrm{GW}} = 60\%$ at $T_{\mathrm{H,GW}} = 15ms$. To sum up, the gateway utilization should be less than 60%.

C. Buffer Capacity

Until now, unbounded input and output queues were used in the gateway. This is only possible for a simulated system. Every system has limited resources and hence a bounded buffer capacity leading to message loss. If the queue is full and a message arrives, the incoming or a queued message must be dropped. The dropping of the incoming message reduces the mean waiting time since this message would get a long

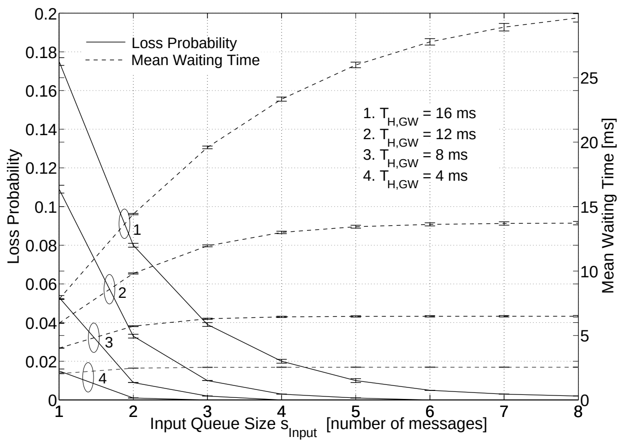


Fig. 3. The loss probability and the mean waiting time in the input queue with different queue sizes s_{Input} and gateway holding times $T_{\text{H,GW}}$.

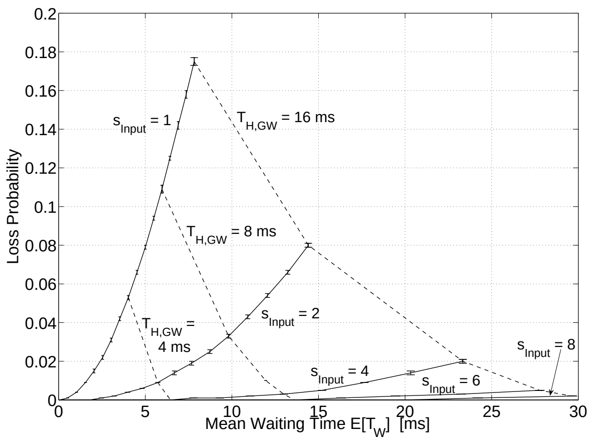


Fig. 4. The mean waiting time in the input queue depending on the loss probability.

waiting time due to the full (long) queue. Another strategy is to drop the message which is at the head of the queue. This also reduces the mean waiting time, since the message in the queue which has the longest waiting time so far is lost. It is easy to implement with a ring buffer. There is nothing more to do than override the oldest message. In further studies, we use the drop head strategy.

1) Input Queue: In Figure 3 both the loss probability and the mean waiting time in the input queue with different queue sizes s_{Input} and gateway holding times $T_{\text{H,GW}}$ are shown. The solid curves show the loss probability and the dotted curves the mean waiting time in the gateways' input queue. Each curve of this figure shows either the loss probability or the mean waiting time for a constant gateway holding time and varying queue size. Reducing the buffer capacity increases the loss probability, but also decreases the mean waiting time.

Figure 4 shows the same data from another point of view. Each solid curve of this figure shows the loss probability and mean waiting time for a constant queue size while varying the gateway holding time. Following a curve from left to right means increasing the gateway holding time without changing the queue size. Each dotted curve connects the points with the same gateway holding time and thus allows a better comparison of the different queue sizes. This figure helps in dimensioning an embedded gateway. Starting with the requirements for the loss probability and waiting time the correct input queue size and optimal gateway holding time can be estimated by means of this figure.

If the embedded gateway is powerful enough, there is no loss in the input queue even if a buffer capacity of one is used. In this case the gateway must handle all incoming messages before the next message arrives, so

$$T_{\text{H,GW}} \leq \min\left\{\frac{T_{\text{H,CAN}_{\text{i}}}}{n}\right\}, \qquad (5)$$

where n is the number of connected domains and $T_{\text{H,CAN}_{\text{i}}}$ is the bus holding time of the i-th domain.

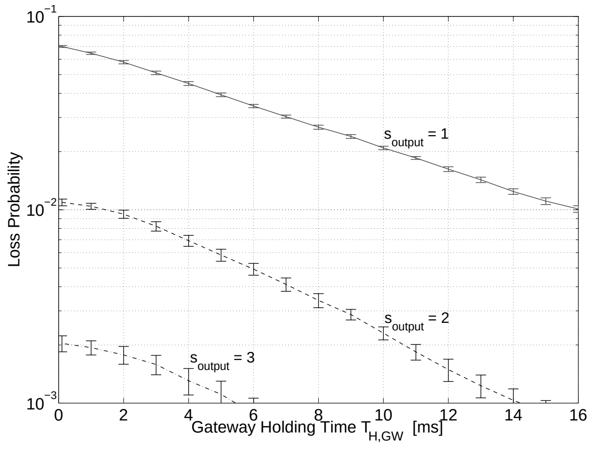


Fig. 5. The loss probability in the output queue depending on the gateway holding time $T_{\text{H,GW}}$.

2) Output Queue: In this scenario the loss probability in the output queue is considered. Figure 5 shows the loss probability in the output queue for different queue sizes s_{Output} depending on the gateway holding time $T_{\text{H,GW}}$. An increasing gateway holding time reduces the loss probability. This is especially true for a buffering size of $s_{\text{Output}} = 1$.

If a short gateway holding time of $T_{\text{H,GW}} = 1ms$ is used, the gateway will push messages into its output queue as fast as they could be transmitted in the destination domain. Due to their low priority they have to wait until all higher priority messages are handled. While waiting for bus access, they are overtaken and replaced by the next message. Thus, the probability for message loss is high.

If a longer gateway holding time (e.g., $T_{\text{H,GW}} = 16ms$) is used, then the messages have a higher chance to gain bus access before the next message arrives. As an interesting effect, the gateway holding time is important for the loss probability in the output queue. Increasing the output queue

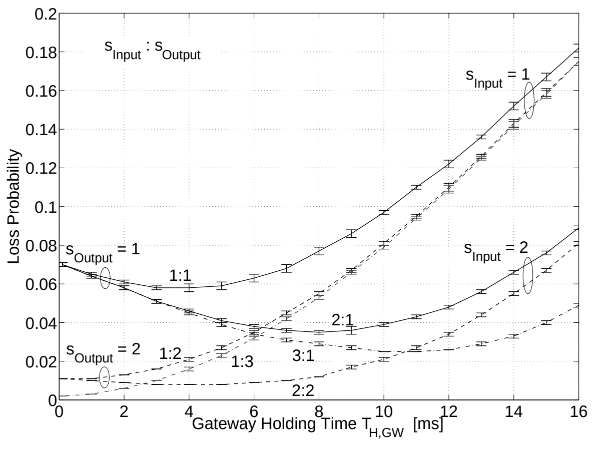


Fig. 6. The loss probability with different queue sizes s_{Input} and s_{Output} depending on the gateway holding time $T_{\text{H,GW}}$.

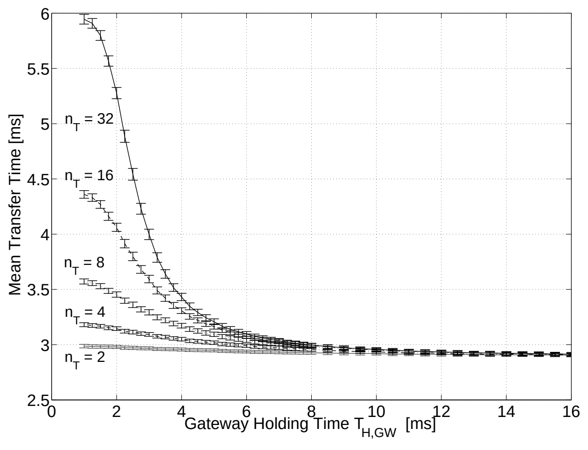


Fig. 7. The mean transfer time of the low priority messages in the destination domain depending on different burst sizes and gateway holding time $T_{\text{H,GW}}$.

size reduces the loss probability. There is no more significant loss if a buffering size of $s_{\text{Output}} = 4$ is used.

3) Entire Gateway: We have considered the input and the output queue individually. Now, the entire system with bounded input and output queues is investigated. The loss within the gateway is naturally the sum of the input and output loss. Above was shown that a powerful gateway reduces the loss in the input queue. The previous section showed that an inefficient gateway reduces the loss in the output queue.

Additionally, the loss probability also depends on the queue sizes. The input and output queues have to share the buffer capacity of the CAN module. We denote the sum of the input and output queue size by $s_{\sum}$. A good partitioning takes the gateway holding time into account. This section shows the optimal partitioning and gateway holding time for a limited buffering size of $s_{\sum} \leq 4$.

Figure 6 shows the loss probability with different queue sizes s_{Input} and s_{Output} for all possible configurations. The minimal loss probability is achieved in the case of $T_{\text{H,GW}} \leq 2ms$ and as much buffer capacity as possible at the output queue ($s_{\text{Output}} = 3$). In the case of a powerful gateway the loss in the output queue dominates the overall loss. There is an optimal partitioning of the buffer capacity for each gateway holding time. The more powerful the embedded gateway, the more buffer capacity must be assigned to the output queue.

There is also an optimal gateway holding time with minimal loss for each partitioning. Using one input and output buffer reduces the buffer capacity to a minimum. In this case the optimal gateway holding time is $3ms \leq T_{\text{H,GW}} \leq 4ms$ as shown in Figure 6.

VII. Talkspurt-Silence Source

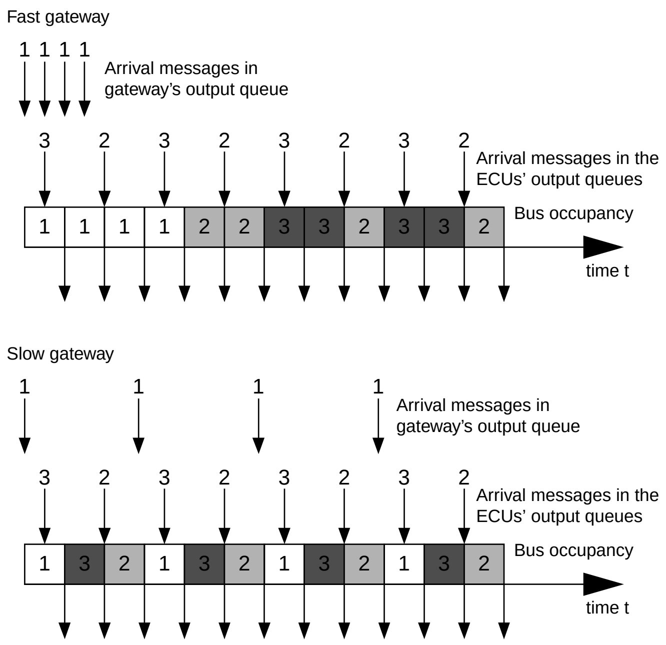
In section VI we described a very conservative dimensioning of the embedded gateway to reduce hardware costs. In a worst case szenario a burst of high priority messages can occur when several ECUs are sending more or less at the same time asynchronously. An intuitive approach to handle the burst would be the usage of a more powerful gateway which leads to higher costs. Therefore, the impact of the gateway holding time on a burst of high priority messages will be investigated.

We model this effect by using one source, sending highest priority messages, which groups this ECUs. In order to investigate this burstiness, we assume a talkspurt-silence source which generates messages with the same mean of $25ms$ and vary the burst size. The talkspurt-silence model alternates talkspurt (on) and silence (off) periods and complies with an on/off process. Only during the talkspurt period n_T packets arrives.

For messages of the bursty source the highest priority in both domains is used. Thus, the buffer capacity of the gateway can easily be limited without loosing messages. Using an input queue size larger than the maximum burst size will avoid message loss, independent of the gateway holding time. The same holds for the output queue. But the output queue size can be further reduced by a proper dimensioning of the embedded gateway. Since a highest priority message gets bus access immediately after the last message is finished, an output buffer capacity of one is sufficient if the gateway holding time is larger than the bus holding time.

Clearly, a bursty source with high priority messages will affect both the start and the destination domain. Due to the lower transfer rate of the destination domain this impact is unacceptable. The objective of this section is to show that a proper dimensioning of the gateway holding time can limit this impact.

Figure 7 shows the mean transfer time of the low priority messages in the destination domain depending on different burst sizes and gateway holding times. For all curves the same mean interarrival time, but a different burst size, was used. Figure 7 shows that a long burst duration increases the mean transfer time of the low priority messages. The figure also shows that this impact on the destination domain can be reduced by increasing the gateway holding time. There is no more significant difference of the curves if a gateway holding



Fast gateway
1 1 1 1
Arrival messages in gateway's output queue
3 2 3 2 3 2 3 2
Arrival messages in the ECUs' output queues
1 1 1 1 2 2 3 3 2 3 3 2
Bus occupancy
time t
Slow gateway
1 1 1 1
Arrival messages in gateway's output queue
3 2 3 2 3 2 3 2
Arrival messages in the ECUs' output queues
1 3 2 1 3 2 1 3 2 1 3 2
Bus occupancy
time t

Fig. 8. Two different sequences of messages on the dependency of the gateway holding time $T_{\mathrm{H,GW}}$.

time of $T_{\mathrm{H,GW}} \geq 10ms$ is used.

A powerful gateway forwards a burst directly into the destination domain. In the destination domain all other messages have to wait until the burst ends due to its high priority. If this burst has finished a burst of messages with the next highest priority will follow. This continues until all except the low priority messages are handled. After some time the bus reaches the stable state again. This leads to a high transfer time for the low priority messages.

Increasing the gateway holding time $T_{\mathrm{H,GW}}$ reduces the mean transfer time of the low priority messages. Such a gateway behaves like the leaky bucket algorithm which is used for traffic shaping and thus leads to a more constant flow of messages. There are no bursts of high priority messages anymore. After a high priority message drops into the destination domain there is enough time to recover before the next high priority message appears.

Figure 8 shows the impact of the gateway holding time $T_{\mathrm{H,GW}}$ on messages sending from other ECUs. This gateway forwards the high priority messages to the destination domain. In the upper part a fast gateway and in the lower part a slow gateway is used. Both parts also show the generation of messages by other ECUs. To compare the two parts these messages are generated equally. The upper part of Figure 8 illustrates how the low priority messages are delayed due to the burst of high priority messages. The fast gateway forwards the high priority messages faster than they can be transmitted in the destination domain. Thus, those messages must be queued in the gateways' output queue. Due to their low priority the messages of the other ECUs must be queued, too. The lower part shows the effect of a slower gateway with a gateway holding time of $T_{\mathrm{H,GW}} = 3 \cdot T_{H,CAN_{125}}$. In this case the burst is mitigated by the slow gateway and thus the low priority messages can be sent before the next high priority message occurs.

So far, only the mean transfer time of the low priority messages was considered. But there are other issues caused by bursts of high priority messages. During these burst all other messages have to be queued in the ECUs. Thus, each CAN module must have a sufficient large output queue to avoid the loss of messages. After this high priority burst there will be a burst of the next priority. The receivers of this messages must be able to handle such bursts.

VIII. Conclusion

Motivated by the limited resources of an embedded CAN-CAN gateway and keeping the costs low we investigated the optimization of resource dimensioning of such a gateway, focused on the gateway processing time and its buffer capacity. Given the fact that CAN systems are often segmented into various domains the gateway performance in terms of load, delay, and message loss plays an important role. In the case of connecting domains with different transfer rates resource dimensioning regarding output and input buffer capacity is more important.

While dimensioning an embedded gateway we always have to find the optimal balance between the transfer time and its loss probability. But in the priority-based network access method of CAN it is more difficult due to the fact that the transfer time and loss probability depends on the traffic characteristic of other ECUs. While dimensioning one ECU and its resources we have to consider all other ECUs.

The minimal loss probability is achieved in the case of a powerful gateway and as much buffer capacity as possible at the output. As an interesting effect, the longer the processing time within the gateway, the more buffer capacity must be assigned to the input queue. Reducing the buffer capacity reduces the transfer time, but increases the loss probability. The gateway holding time is critical if a burst of high priority messages occurs. A powerful gateway forwards bursts from one domain directly to the other. A longer holding time leads to a more constant flow of messages and thus reduces the impact, mainly in low-speed CAN domains.

For this paper we have used simple traffic models. However, part of our ongoing work is to refine these models.

Furthermore, we used the Round Robin scheduler in the gateway for serving the incoming messages of the connected domains. Part of our ongoing work is to investigate different scheduling algorithms, e.g., the *Weighted Round Robin* for preferring connected domains with higher transfer rates or a priority-based algorithm.

We also used a fixed assignment of the available buffer capacity. A dynamic allocation could reduce the loss probability due to a better utilization and partitioning of the available buffer capacity.

In many fields of application, different network technologies with completely different medium access methods are interconnected. In the long term, our studies will be extended to heterogeneous interconnection scenarios with different access mechanisms and transfer rates, e.g., a CAN-Ethernet gateway.

Acknowledgment

The authors would like to thank Lars Burgstahler and Volker Feil for many fruitful discussions and for their contribution to [12]. The authors would also like to thank the reviewers who have read and helped to improve the presentation of this paper.

References

[1] E. W. Biersack, "Annotated bibliography on network interconnection," *IEEE Journal on Selected Areas in Communications*, vol. 8, no. 1, pp. 22–41, January 1990.

[2] H. Ekiz, A. Kutlu, and E. Powner, "Design and Implementation of a CAN / CAN Bridge," *ispan*, pp. 507–513, 1996.

[3] ——, "Implementation of CAN/CAN bridges in distributed environments and performance analysis of bridged CAN systems using SAE benchmark," *Engineering New Century, Proceedings. IEEE*, pp. 185–187, April 1997.

[4] J. Eltze, "Double-CAN Controller as Bridge for Different CAN Networks," in *Proceedings of the 4th International CAN Conference*, Erlangen, Germany, 1997.

[5] K. Etschberger, *Controller Area Network*. Fachbuchverlag Leipzig, 2002, in German language.

[6] Infineon Technologies, *XC167-16 – 16-Bit Singe-Chip Microcontroller with C166SV2 Core – Volume 2(of 2): Peripheral Units*, April 2004, User's Manual, V2.0.

[7] Institute of Communication Networks and Computer Engineering, "IKR Simulation Library," Stuttgart, Germany, 2006. [Online]. Available: http://www.ikr.uni-stuttgart.de/Content/IKRSimLib/

[8] International Organization for Standardization, *ISO 11898-1:2003 – Road vehicles – Controller area network (CAN) – Part 1: Data link layer and physical signalling*, November 2003.

[9] L. Kleinrock, *Queueing Systems - Volume 2: Computer Applications*. Wiley-Interscience, 1976.

[10] Robert Bosch GmbH, *CAN Specification Version 2.0*, September 1991.

[11] R. Rüdiger, "Prioritätswartesysteme fr die Modellbildung von CAN-Systemen," Fachbereich Mathematik und Technik, FH Bielefeld," 3. Norddeutsches Kolloquium ber Informatik an Fachhochschulen, May 1998, in German language.

[12] J. Sommer, L. Burgstahler, and V. Feil, "An Analysis of Automotive Multi-Domain CAN Systems," in *Proceedings of the 12th EUNICE Open European Summer School*, September 2006.

[13] M. Stuempfle and J. Charzinski, "Simulation of Heterogeneous CAN-Systems," in *Proceedings of the 2nd International CAN Conference*, London, UK, 1995.

[14] W. Zhang, M. S. Branicky, and S. M. Phillips, "Stability of networked control systems," *IEEE Control Systems Magazine*, vol. 21, no. 1, pp. 84–99, February 2001.

[15] R. Zurawski, *The Industrial Information Technology Handbook*. CRC Press, 2004.