An Analysis of Automotive Multi-Domain CAN Systems

Jörg Sommer[1], Lars Burgstahler[2], Volker Feil[2]

[1]University of Stuttgart, Institute of Communication Networks and Computer Engineering
Pfaffenwaldring 47, 70569 Stuttgart, Germany
Email: joerg.sommer@ikr.uni-stuttgart.de
[2]DaimlerChrysler AG, Vehicle IT and Services
HPC 050/G021, 70197 Stuttgart, Germany
Email: {lars.burgstahler, volker.feil}@daimlerchrysler.com

***Abstract*— In today's passenger-cars, a large number of electronic control units (ECUs) and functions are distributed over a multitude of bus systems (domains), connected via gateways. What was once thought of as a means to reduce complexity and facilitate management has become a new challenge with the increasing number of cross-domain functionalities, i.e. applications that exchange data over domain boundaries. Gateways often become bottlenecks between buses (e.g. by increasing latency) and in the case of Controller Area Network (CAN) buses, they can pose further problems with respect to the priority-based network access method.**

Therefore, the introduction of cross-domain functions increases the necessity for a thorough performance evaluation of the vehicles' network architectures during the concept and development phases.

In this paper we analyse the impact of cross-domain functionalities on automotive CAN system's performance, especially with respect to delay. The focus of our work is on modelling, simulation, and analysis of the system. Therefore, we describe our models of ECUs, CAN buses, and gateways and how we used them to simulate a complete multi-domain CAN system. Based on this, the results of performance analysis within the gateway and the domains are presented.

I. Introduction

Since the introduction of *Controller Area Network* (CAN) systems into vehicles by Mercedes-Benz in 1992, the number of interconnected *Electronic Control Units* (ECUs) in passenger cars has tremendously increased from less than ten to more than 70 in today's upper-class cars. Shortly after introducing the first CAN system, engineers already implemented physically separated CAN systems (domains) in a car, connected via gateways. Today's upper-class cars typically have between 5 and 6 CAN systems.

The reasons for multi-domain systems were manifold: vehicle domains with different emphasis on safety and reliability (i.e., powertrain, chassis, comfort, and telematics) needed to be kept separate. Design, development, and maintenance were facilitated by grouping associated functionalities on one bus system. One of the main targets was to reduce complexity. Furthermore, in case one bus fails, other buses were not affected[1]. Then, an increasing number of ECUs and functions generated more and more traffic, thus scalability and easier administration and management were ensured by separated buses. ECUs were distributed among the buses in a way to keep cross-domain traffic as low as possible. The functions were distributed among the ECUs in a way to keep overall traffic (even within a domain) as low as possible.

Today's situation is characterised by a stagnating number of ECUs. Although manufacturers try to limit the number of functions to keep the complexity manageable, the necessity to install innovative features to distinguish cars from the competitors' products (e.g., safety or comfort features) or to satisfy new legal regulations, nevertheless leads to an increasing number of functions.

Many functions affect components in different domains of the vehicle. A classical example is *speed-controlled volume* (SCV): The wheel rotations per minute are measured by a sensor attached to an ECU in the chassis domain. This ECU calculates the speed and passes it on through one or two gateways (i.e., via one or two domains) to the audio tuner which then adapts the volume – low at low speed, high at high speed. A future novel feature could be map-based car dynamics: the in-car navigation extracts information about road bends or altitude change from the map and passes the information (again over gateways) on to the chassis or powertrain domain respectively [2]. There, the information can be used to support air suspension and braking (by increasing stored pressure before the bend) or gear shifting (before losing momentum at a hill). Finally, vehicle state information is visually and/or acoustically displayed in today's vehicles. Display usually takes place in the telematics/entertainment domain, whereas the sources of the information are distributed throughout the other domains. On the other hand, many functionalities located somewhere in the car can be controlled via some central controller (e.g., a joystick-like device) located

[1]Some manufacturers deliberately implement buses containing only the ECUs that are most likely to be destroyed in case of an accident. Thus, the vehicle's more important buses still remain operational, and the vehicle can be moved out of the danger zone.

in the telematics/entertainment domain.

As a result, the separation of the domains will vanish even more and traffic through gateways will increase. One of the biggest issues is low-priority traffic from the telematics domain. This traffic, originating from a Media Oriented Systems Transport (MOST) ring where bandwidth and delay are hardly issues, can be severely impaired by priority-based arbitration on CAN buses and in gateways. Furthermore, its behaviour might differ from what we know so far in a vehicle. With respect to the characteristics of CAN buses new challenges have to be considered to master the impacts of this development. Instead of becoming less complex, vehicle networks tend to become more complex.

The remainder of this paper analyzes the impact of low-priority traffic in automotive CAN systems with emphasis on cross-domain issues. First, a description of a multi-domain CAN system and its components is given. Then, in section three the modelling of these components which will be used in a simulation is described. We will further explain the complete simulation model. Section four shows the results that have been achieved with the simulation regarding some typical effects of high-volume, low-priority traffic. Mainly, we investigate the mean values for waiting time and delay depending on domain utilisation. Finally, a conclusion and an outlook is given in section five.

II. Multi-domain CAN system

In this section we describe the automotive multi-domain scenario that we evaluated. Real automotive CAN systems consist of several buses. Some topologies are star shaped with one central interconnection device, while others consist of buses interconnected in different patterns. Our example network represents the general interconnection scheme in its simplest form by using three buses and one gateway.

In the following we will briefly describe the components of the network. In the next section, our modelling approach to each component will be laid out.

A. Electronic Control Unit

An ECU is an embedded system for controlling one or more of the electrical subsystems in a vehicle. Examples for ECUs are the engine controller module, the door module (windows, opener), and the heating, ventilation, and air conditioning module.

Most ECUs in a vehicle are attached to a CAN bus. Additionally, sensors, actuators, and even subbuses (e.g., *Local Interconnected Network* (LIN)) are connected to the ECUs to provide information, execute actions or handle subtasks. An ECU contains a 8-bit to 32-bit microprocessor, input/output interfaces to the CAN, sensors and actuators and sometimes subbuses, and memory (RAM, ROM, Flash).

Based on information from any input interface, e.g., sensors, the ECUs determine parameters for the actuators. Today, no single ECU is isolated; they are interconnected and exchange large amounts of data, i.e., information from sensors or parameters to actuators.

B. CAN Bus

CAN was developed in the 1980s by Robert Bosch GmbH, a German automotive supplier, as a communication bus for in-vehicle electronics. Bosch's original specification [10] was submitted for international standardization in 1991 and later became an ISO standard documented in [6].

CAN is a robust, low-cost, and simple event-triggered technology. It uses *Carrier Sense Multiple Access with Collision Detection and Non-Destructive Bit Arbitration* (CSMA/CD-NDBA) as medium access control mechanism. Simultaneous access of several nodes to the bus is detected and NDBA resolves the collision in such a way that the message with the highest priority wins the arbitration process unmodified. This means that the transmission of the message continues after the arbitration and is not restarted — no arbitration delay is introduced. CAN is non-preemptive in the sense that a message that won the arbitration process will always be transmitted completely without regard to higher priority signals that arrive at a later point. Due to its priority scheme and the sometimes stochastic arrival process of higher priority messages, CAN can not guarantee deterministic response times for messages with lower priorities.

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 length of 11 bit (Standard CAN) or 29 bit (Extended CAN) respectively. An 11 bit identifier means that up to 2048 different identifiers are available on a bus. Each identifier relates to a message with a certain content, e.g., wheel rpm, selected gear, head light state, etc. 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 broadcast. CAN uses a bit-stuffing mechanism for synchronisation and error detection which increases the frame length and reduces the throughput [9].

The CAN specification limits the maximum bitrate to 1 MBit/s. Typical bitrates in automotive environments are 500 KBit/s and 125 KBit/s. More detailed descriptions of CAN can be found in [3] and [14].

C. Gateway

Gateways are a special kind of ECU that connect two or more buses and pass data from one bus to others. Passing data includes simple message forwarding as well as assembly of new messages from data of received messages before forwarding. Some car manufacturers use special gateway ECUs with no more functionality than needed for the interconnection, while others add the gateway functionality to an existing ECU that is already connected to several domains.

Gateways introduce additional delay due to message processing: the target domain has to be identified, sometimes only certain signals within a message need to be forwarded. Thus, messages are dissected and signals are reassembled in

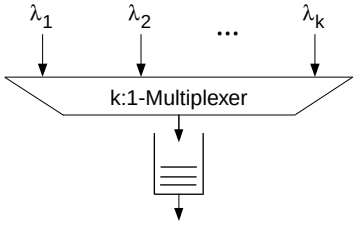

Fig. 1. Model of an ECU with a *k:1* multiplexer and one output buffer for all messages.

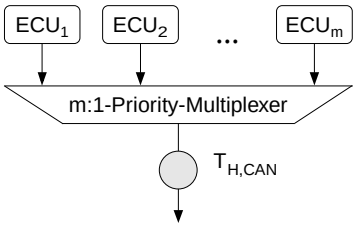

Fig. 2. Model of a CAN bus with *m* ECUs, a *m:1* priority multiplexer, and a phase representing the CAN bus.

a new message and finally access to the new bus costs time because arbitration must be won first.

If the connected buses have different transfer rates, the gateway must provide enough buffering capacity in the direction from the fast to the slow bus, otherwise message loss can occur. In the opposite direction, some buffer is also necessary, to store messages with low priority until they win the arbitration and can be sent.

To sum up, gateways are a necessity producing additional costs and weight, while not directly providing a customer-observable benefit. Hence, it is all the more important to optimise their performance, while minimising their disadvantages.

III. Modelling

A. Electronic Control Unit

An ECU can be modelled as a sender and a receiver part [13]. Messages originating in the ECU are buffered in the sender part. Incoming messages from the CAN are buffered in the receiver part.

Usually an ECU sends different messages with different identifiers. Depending on the implementation of the sender [4], either a buffer is provided for each message identifier, or a single buffer for all identifiers is used (Figure 1). The different receiving buffer characteristics are mentioned in [4], [13] in more detail.

The ECU model in Figure 1 sends messages with k different identifiers. These messages can be sent periodically (approximately constant interarrival time) or sporadically with the mean arrival rate λ_i and $i = 1, .., k$. In our model, queues with a FIFO discipline are used for the sender and the receiver parts, but other queuing disciplines are possible as well.

B. CAN Bus

Models of CAN buses were presented in many previous papers ([11], [12], and [13]). As shown in Figure 2, we use a single non-preemptive server as a model of the bus [13]. The bit 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 $T_{\mathrm{H,CAN}}$ and/or a suitable distribution function. The priority multiplexer simulates the arbitration of the CAN by always selecting the queue with the highest message identifier at its head first. Thus, the message identifier is equivalent to the priority.

$\mathcal{E}_i$ denotes the set of message identifiers which are sent from ECU_i. Within a domain the sets $\mathcal{E}_i$ are pairwise disjunctive

$$\mathcal{E}_\mathrm{i} \bigcap \mathcal{E}_\mathrm{j} = \emptyset \quad \text{for } i \neq j. \tag{1}$$

In Figure 1 $\mathcal{E}_1$, $\mathcal{E}_2$ ... $\mathcal{E}_k$ are associated to $\lambda_1, \lambda_2, \ldots \lambda_k$.

C. Gateway

With respect to CAN buses, two types of gateways exists: (1) gateways which connect two or more CAN buses, and (2) gateways which connect CAN buses to buses with different technologies [8], e.g. MOST or LIN. Although we only consider the first case in this work, the model shown in Figure 3 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, protocol adaption, and must possibly adapt to QoS requirements.

Our gateway model shown in Figure 3 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)*.

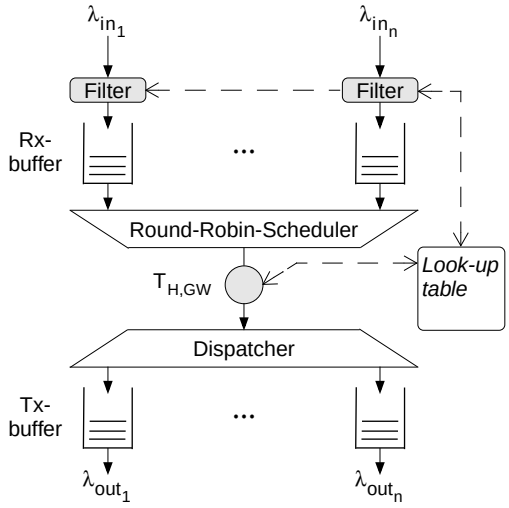

Fig. 3. Model of a gateway with filter, receiving queue, server for identifier translation, and transmission queue.

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 and to which one, 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. Hence, to simplify the model 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 on 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 destination identifier of an incoming message can be altered by means of the look-up table (this corresponds to message recomposition). The gateway can also change the identifier of a message and so their priority.

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 complete transfer time of an individual message in the gateway can be calculated as

$$T_{\mathrm{GW}} = T_{\mathrm{Rx}} + T_{\mathrm{H,GW}} + T_{\mathrm{Tx}}. \tag{2}$$

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 a outgoing message can be send in the destination domain.

D. Complete Model

The complete communication systems architecture consists of a number of ECUs connected to different buses. These compounds build the domains which are interconnected by gateways. Besides the topology, the message arrival process at the ECUs is the primary factor that influences the overall behaviour and performance of the architecture.

Altering the model, i.e., restructuring the topology, moving the ECUs from one domain to another or mapping messages to different originating ECUs permits to compare and evaluate complex architectural variations or additional components. Non-pervasive measurement methods, i.e., methods within the simulation environment that do not influence the simulated process itself, are used to measure delay, rate, queue length etc. These results help to dimension and optimise an architecture with respect to given requirements.

IV. Performance Evaluation

The model of section III has been implemented with the *IKR Simulation Library* (IKRSimLib) [5], an object-oriented class library for event-driven simulation.

In our simulation the CAN messages have a constant payload length of 8 bytes and bit-stuffing is not implemented. Bit-stuffing can increase the bus load by up to 20%.

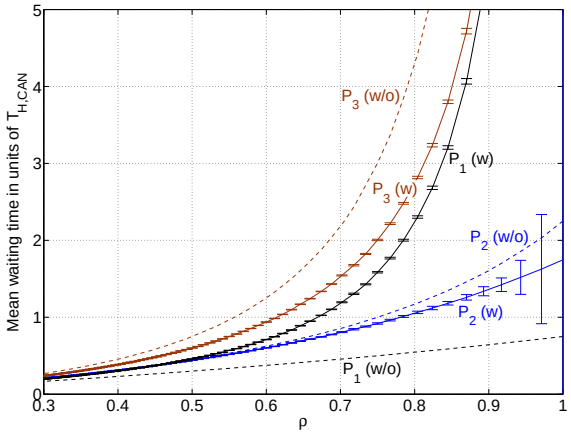

Fig. 4. The impact of the head-of-line blocking effect on the mean waiting time.

A. Head-of-Line Blocking

When messages at the head of a FIFO queue cannot be served due to their low priority (e.g., at the output queue of a CAN gateway), higher priority messages behind them cannot be served either. The FIFO discipline prevents high-priority messages from overtaking low-priority messages that lie ahead in the queue. This effect is called *head-of-line blocking*.

To analyse this effect, a scenario with 3 priority classes is used. The messages with priority 1 and 3 (P_1 and P_3) are sent from ECU_1 and messages with priority 2 (P_2) are sent from ECU_2 ($\mathcal{E}_1 = \{1, 3\}$ and $\mathcal{E}_2 = \{2\}$). Since messages with P_2 always win arbitration against messages with P_3, waiting P_3 messages can block P_1 messages in ECU_1 although P_1 messages would always win the arbitration process.

All three message types are generated by a Markovian arrival process and they have equal mean arrival rates ($\lambda_i = \lambda_j$ and $i, j = 1, 2, 3$). The bus utilisation ρ results from the sum of the arrival rates λ_i of the different messages.

In Figure 4 the head-of-line blocking effect is shown. The dotted curves show the mean waiting time of P_i messages ($E[T_{W,i}]$) without head-of-line blocking. $E[T_{W,i}]$ can be calculated as follows [7]:

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

with

$$\varrho_i = \sum_{j=1}^{i} \rho_j,$$

$$\text{and } \rho_j = \lambda_j \cdot h,\ h = E[T_{\mathrm{H,CAN}}],\ \rho_0 = 0. \tag{4}$$

ρ_j $(j = 1, 2, 3)$ is the bus utilisation generated by P_j messages. As expected, the mean waiting time for lower-priority messages is higher than for high-priority messages and waiting time in general increases with increased bus load.

The solid curves in Figure 4 display the simulation results for the waiting time, including head-of-line blocking. We can

see (as expected) that P_2 messages still wait shorter than P_3 messages. Yet, they also wait shorter than P_2 messages without head-of-line blocking. The reason is that P_1 messages are blocked by P_3 messages so often that P_2 messages have a higher probability to win arbitration. Since arrival rates of P_1 and P_3 messages are equal, the blocking occurs so often that P_1 messages are waiting even longer than P_2 messages.

There are several solutions to decrease head-of-line blocking: If bus utilisation is sufficiently low, blocking will rarely occur. Another option is to decrease the arrival rate of lower priority messages. Both approaches are not always feasible. In a *drop-head-queue* the message at the head of the queue will be deleted after a certain time, thus removing the cause of the blocking. However, the loss probability of low-priority messages increases.

To avoid head-of-line blocking completely, each priority needs its own queue within the ECU. Queues with a priority discipline yield the same result: Messages are sorted by priority as they enter the queue. Unfortunately, these solutions are often too slow or too expensive.

An aspect that has not been dealt with in this section is the *Coefficient of Variation (CoV)* of the low-priority traffic. Results from the simulations suggest that a large CoV might have an impact on the traffic with a higher priority. We cannot discuss this effect in this paper, because further research is needed.

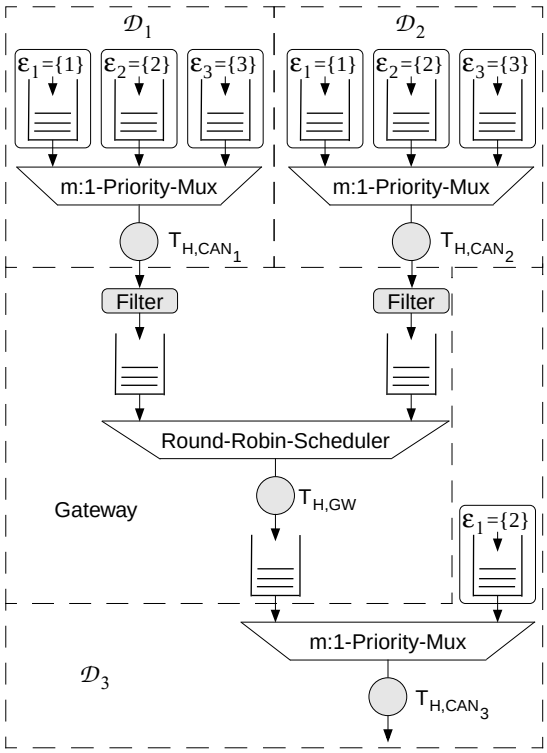

Fig. 5. The cross-domain scenario with three domains connected through a central gateway.

B. Cross-domain traffic

We have described the head-of-line blocking in detail because it plays an important part in cross-domain traffic. Bus arbitration works perfectly fair on one bus. However, at a gateway's output, all messages of one bus wait in line to enter another bus. Suddenly high-priority messages that were successful on their own bus could be blocked by low-priority messages from the same or another bus which cannot enter the destination bus. This has a significant impact on the end-to-end delay of messages (T_{E2E}) which is calculated as follows:

$$T_{\mathrm{E2E}} = T_{\mathrm{W,ECU}} + \sum_{i=1}^{n} T_{\mathrm{H,CAN_i}} + \sum_{i=1}^{n-1} T_{\mathrm{GW_i}}. \qquad (5)$$

The waiting time within an ECU until arbitration is won ($T_{\mathrm{W,ECU}}$) depends on the messages' priorities and the bus load. Transmission and transport times (aggregated in $\sum_{i=1}^{n} T_{\mathrm{H,CAN_i}}$) are constant and given by technology and topology. Hence, the processing time in gateway ($\sum_{i=1}^{n-1} T_{\mathrm{GW,i}}$) is the single point that can cause significant delay variations.

For this to be effective, the gateway in our simulation model has only one output queue with a FIFO discipline per bus. This corresponds to the real implementation in most vehicles' gateways.

Our model contains three domains ($\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$) of equal bandwidth ($T_{\mathrm{H,CAN_i}}, i = 1, 2, 3$) as shown in Figure 5. $\mathcal{D}_1$ and $\mathcal{D}_2$ are set up identically. Each one contains three independent sources with P_1, P_2, and P_3 respectively. The load generated by sources 1 and 3 is $\rho_i = 0.1,\ i = 1, 3$. The overall utilisation in all three domains depends on ECU_2's load which can be varied. $\mathcal{D}_1$ and $\mathcal{D}_2$ are connected to $\mathcal{D}_3$ through a single central gateway. Cross-domain traffic consists of high-priority messages from $\mathcal{D}_1$ (P_1) and low-priority messages from $\mathcal{D}_2$ (P_3) into $\mathcal{D}_3$. In $\mathcal{D}_3$ there is only one single source with a medium priority (P_2). Low-priority messages from $\mathcal{D}_2$ will lose arbitration again and again, thus blocking P_1 messages from $\mathcal{D}_1$ at the gateway's output.

Figure 6 shows the mean end-to-end delay of P_1 messages from $\mathcal{D}_1$ and P_3 messages from $\mathcal{D}_2$. Figure 7 shows the respective transfer times in the gateway (waiting and processing). As expected, P_3 messages take more time from source to destination. In case of a low bus utilisation ($\rho \leq 0,50$), the end-to-end delay difference between low- and high-priority messages is negligible. However, with a higher bus utilisation, the P_3 messages experience an increasingly higher delay. For $\rho \approx 0.7$, P_3's delay is about 1.8 times higher, for $\rho \approx 0.8$, the factor is already 2.4.

As shown in (5), the influencing factors of delay are waiting time in the ECU and waiting time in the gateway because they are variable. If we compare the results from Figures 6 and 7, we see that the end-to-end delay of P_1 messages corresponds almost exactly to their transfer time in the gateway plus a constant. First, this means that the waiting time in the source ECU and arbitration delay to $\mathcal{D}_3$ (once the P_1 message is at the head of the queue) remains almost constant. This is obvious because these messages always win the arbitration,

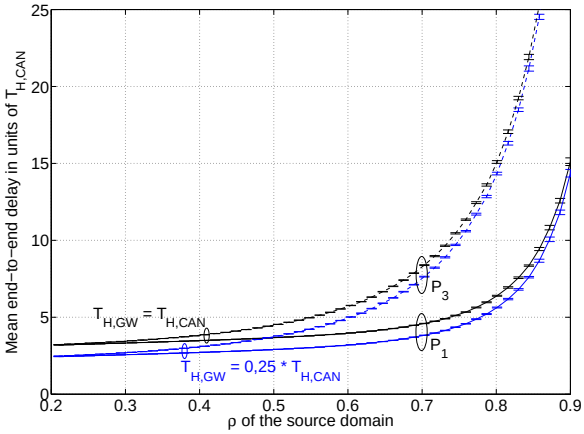

Fig. 6. End-to-end delay of P_1 and P_3 messages with different gateway service times $T_{\mathrm{H,GW}}$.

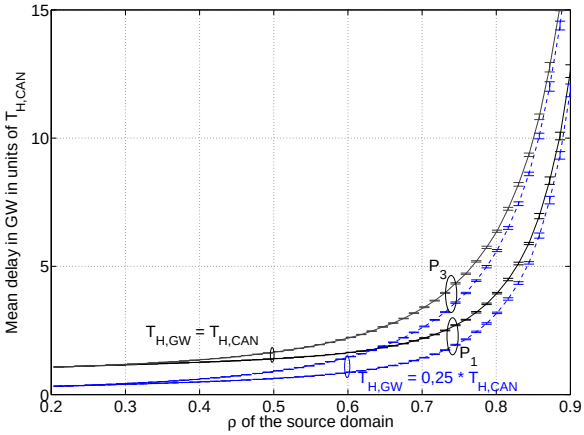

Fig. 7. Mean transfer time of P_1 and P_3 messages in gateway with different gateway service times $T_{\mathrm{H,GW}}$.

no matter how high the load is. High load cannot cause additional delay larger than the transmission duration of one message[2]. Secondly, it shows that the higher delay is almost exclusively caused by head-of-line blocking in the gateway. This also explains why the transfer time for P_1 messages in the gateway is almost as high as for P_3 messages, because their share of traffic is the same. Thus, each P_1 message has a high probability to encounter an already waiting P_3 message. However, once at the head of the queue, their arbitration waiting time is much shorter.

As for the P_3 messages, we can see that the end-to-end-delay is roughly twice the transfer time in the gateway. This is reasonable because the load in all domains is equal and these messages have to wait equally long to win arbitration in each domain.

[2]The maximum waiting time of one message duration occurs if the P_1 message arrives just after arbitration. Since the probability for arrival at any point during message transmission is equal, the maximum mean waiting time for highest priority messages is half the duration of a message transmission time.

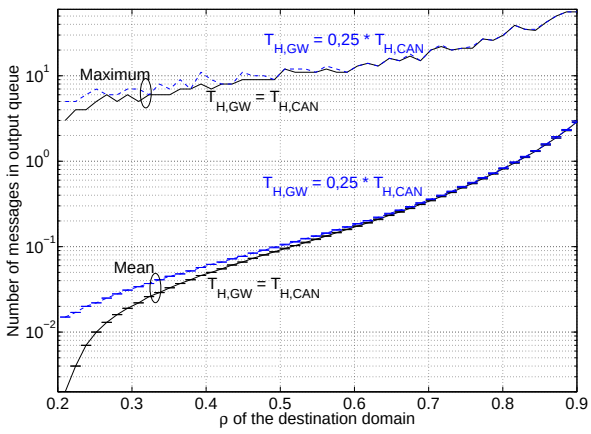

Fig. 8. Mean and maximum number of messages in the gateway's output queue.

Both figures show that the impact of the gateway service time $T_{\mathrm{H,GW}}$ on the end-to-end delay is not significant. Of course, the precondition is that the service time is shorter than the message transmission time. However, a shorter service time is a valuable means to keep the input queues of the gateway empty and to prevent message loss on the source buses. This holds especially for gateways with input queues from multiple domains.

Figure 8 shows another view on the head-of-line blocking effect. The mean queue length at the gateway's output is quite short, yet the maximum values show that significantly more messages are queued under high load conditions. This supports the conclusions drawn before. Thus, even if short queues were sufficient to cover the average traffic, the maximum values show that some reserve in the queue is needed to prevent loss, especially of important high-priority messages. Additionally, we can see that a shorter service time leads to slightly larger output queues, because the messages from the various input queues are faster forwarded to one single output queue (see above), i.e., the input queue length tends toward zero.

C. Cross-domain traffic with a Greedy Source

Normally, control-based communication systems like CAN are not well-suited for streaming high-volume data traffic of innovative applications like map-based car-dynamics (see section I) or browsing through iPod song lists on the instrument cluster (depending on the vehicle's architecture the iPod interface could be on a different bus than the instrument cluster). Streaming is a novel type of data traffic in the context of automotive CAN buses. The data is usually sent with a transport protocol at a very low priority to not influence mission-critical traffic (e.g., for safety features). Recent studies have shown that the impact of this low-priority traffic can indeed be neglected, but this is only due to a rather high minimum interarrival time of transport-protocol messages. As a consequence, the throughput of these applications and

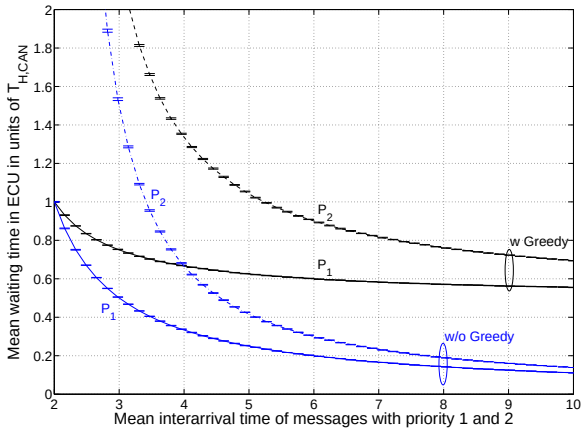

Fig. 9. Mean waiting time of messages in ECU_1 and ECU_2 in dependence of their mean interarrival time.

probably their value to the customer will be very limited.

An obvious solution is to reduce or suspend the minimum interarrival time of transport protocol messages, while keeping their priority low. In our simulation such applications are modelled as so-called *greedy sources*, i.e., sources that can always send a message, if the bus is empty. As a consequence, the bus utilisation reaches a constant $\rho = \sum_{i=1}^{m} \lambda_i h = 1$. The impact of this consequence, with a focus on head-of-line blocking in one single domain, will be evaluated in the following.

To fulfill the requirement of not influencing high-priority messages, the greedy source traffic gets the lowest priority m. Hence, its mean arrival (i.e., sending) rate λ_m decreases if the arrival rate λ_i of any other source increases. This means that the greedy source's amount of traffic and mean sending rate has to be derived from the other sources' offer:

$$\rho_{\mathrm{Greedy}} = 1 - \sum_{i=1}^{m-1} \lambda_i \cdot h, \tag{6}$$

and therefore

$$\lambda_{\mathrm{Greedy}} = \lambda_m = \frac{1}{h} - \sum_{i=1}^{m-1} \lambda_i. \tag{7}$$

The following scenario uses one normal source in ECU_1 and ECU_2 and a greedy source in ECU_3. Figure 9 shows the mean waiting time that messages of ECU_1 and ECU_2 experience in their respective ECUs before they get access to the bus. The x-coordinate shows the expectation of the mean interarrival time $E[T_{A,i}] = 1/\lambda_i$ of messages with P_i. In case of $E[T_{A,i}] = 2$ the bus is fully occupied by messages from ECU_1 and ECU_2 which explains the identical waiting time with or without greedy source at that point.

As an interesting effect, the full occupancy of the bus transforms the system in our model into a time-slotted system with the fixed interval $T_{\mathrm{H,CAN}}$[3]. This happens because the greedy source will send immediately after any message, if neither ECU_1 nor ECU_2 will send at that point. Consequently, the bus is always occupied by equally sized messages. In this case the medium access can be compared to a priority-based asynchronous time-division multiplex access scheme. The mean residual service time $E[T_R]$ in such a system is given by $T_{\mathrm{Slot}}/2$ where T_{Slot} is the duration of a single time slot. For $E[T_{A,1}] = E[T_{A,i}] \to \infty$ and a greedy source, we get an additional mean waiting time for messages of

$$\begin{aligned} \lim_{E[T_{A,i}]\to\infty} \Delta E[T_{\mathrm{W,i}}] &= \\ |E_{\mathrm{Greedy}}[T_{\mathrm{W,i}}] - E_{\mathrm{NoGreedy}}[T_{\mathrm{W,i}}]| &= \\ \frac{T_{H,CAN}}{2} & \end{aligned} \tag{8}$$

The simulation yields appropriate results already for $\rho \leq 0.85$ without greedy and Figure 9 clearly shows this trend for larger interarrival times, too.

Messages from ECU_2 will suffer worse from greedy source traffic. Actually, the lower the priority of a message is (while still being higher than the greedy source's priority), the worse the suffering will be. This is because sooner or later the greedy source will win the arbitration process in a system where $\sum_{i=1}^{m-1} \lambda_i h < 1$. Any message that arrives during the greedy source's message transmission has to wait without respect to its own priority. The waiting message with the highest priority will then win the next arbitration and increase the waiting time of the waiting messages with lower priority even more. Hence, for $E[T_{A,2}] = E[T_{A,i}] \to \infty$ and a greedy source, the mean waiting time of ECU_2 messages will be slightly larger than $T_{\mathrm{H,CAN}}/2$ in the case without a greedy source.

V. Conclusion and Outlook

In this paper we analyse an important point in automotive multi-domain CAN systems: the influence of low-priority messages on messages with a higher priority. Given the fact that modern automotive CAN systems are segmented into various domains and that cross-domain traffic, especially of low-priority messages, is increasing, two issues were of primary interest: the waiting time in the source ECUs and the end-to-end delay (including the transfer time in a gateway) of high-priority messages. In both cases, head-of-line blocking is identified as the reason for probably increased delay. We have developed a model of the CAN system including ECUs, buses, and gateways. Further, a simulation tool was implemented and several scenarios were evaluated.

The results show that under moderate load conditions within a domain (i.e., $\rho \leq 0.5$), low-priority traffic does not seriously influence the high-priority traffic. Under higher load conditions the results depend heavily on how the different messages are deployed on the ECUs. If high- and low-priority messages are not sent from the same ECU, i.e., no head-of-line blocking

[3]Note that in our model each message has a payload of 8 byte. In real systems, the message sizes differ and the intervals are not equal.

occurs within an ECU, the impact is still acceptable. If however, one ECU sends high- as well as low-priority messages, the additional delay for the high-priority messages may be critical. Unfortunately, this issue cannot be easily solved in another way than to keep the overall load rather low.

Some issues are still open to further research and will be evaluated in the next steps. First, we need to identify quantitatively in more detail how the CoV of low-priority traffic changes its influence on high-priority traffic. The latest results of our work lead to the assumption that a certain degree of the CoV may have serious impacts on the high-priority traffic even under moderate load conditions. Then, we have only shown the impact of greedy sources within one domain; the results of studies with several domains are yet to be completed.

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

In the long term, our studies will be extended to other network technologies with different access mechanisms. This work will surely include the MOST technology, which is already used in the infotainment domain of upper-class vehicles. Recently, some car manufacturers have started to introduce FlexRay, a high-speed network technology with a completely different medium access method. While modelling and implementation for MOST is finished and first results have been obtained, our work on FlexRay is still at the beginning.

ACKNOWLEDGMENT

The research described in this paper is supported by DaimlerChrysler AG, Stuttgart.

The authors would like to thank Andreas Reifert for many fruitful discussions and Rainer Blind for his contribution [1] to the simulation tool.

REFERENCES

[1] R. Blind, “Entwurf und Implementierung eines Simulationsmodells zur Analyse von Interdomainverkehr in prioritätsbasierten Nachrichtennetzen,” Semester Thesis, University of Stuttgart, 2006, in German language.

[2] ERTICO - ITS Europe, “MAPS & ADAS,” 2006. [Online]. Available: http://www.prevent-ip.org/

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

[4] H. Hörner, A. Raisch, and O. Meili, “Basis-Software-Komponenten in AUTOSAR – ein solides Fundament,” in *VDI Berichte*, no. 1907, October 2005, pp. 409–419, in German language.

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

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

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

[8] W. S. Levine and D. Hristu-Varsakelis, *Handbook of Networked and Embedded Control Systems*. Birkhäuser, 2005.

[9] T. Nolte, H. Hansson, and C. Norstrom, “Probabilistic Worst-Case Response-Time Analysis for the Controller Area Network,” in *9th IEEE Real-Time and Embedded Technology and Applications Symposium*, May 2003, pp. 200–207.

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

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

[12] S. Schneider, “Performance Analysis for Automotive CAN Systems,” in *Proceedings of the 3rd International CAN Conference*, Erlangen, Germany, 1996.

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

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