

Delay Bounds for CAN Communication in Automotive Applications

Ulrich Klehmet, Thomas Herpel, Kai-Steffen Hielscher, Reinhard German

Department of Computer Science 7
(Computer Networks and Communication Systems)

University of Erlangen-Nürnberg
Martensstraße 3

D-91058 Erlangen, Germany

Email: {klehmet, herpel, ksjh, german}@informatik.uni-erlangen.de

Abstract. As a widespread automotive network, CAN (Controller Area Network) buses are deployed in modern cars to fulfill the demands of more than 90 collaborating electronic control devices. For safety critical applications, fast and reliable data transfer is indispensable, since often hard real-time transmission deadlines have to be met to assure a safe operation of the vehicle. Therefore, deterministic performance evaluation methods are inevitable for the validation of systems that must guarantee hard delay bounds and timeliness of information processing. One recent deterministic modeling approach is Network Calculus, which allows to determine worst case transmission times. Based on real-world communication CAN bus data, we generate appropriate modeling elements for the Network Calculus as arrival and service curves that reflect all priorities of CAN traffic. While the highest priority traffic on a CAN bus is known to have real-time properties, the results of this paper provide closed and easily applicable formulas to determine delay bounds of messages on all priority levels.

1 Introduction

Modern cars are equipped with various ECUs (Electronic Control Units) for entertainment and infotainment purposes, wireless connectivity or enhancements of active and passive occupant protection.

Besides the increasing power consumption, the communication and data exchange between the single control units is one of the most critical tasks to be considered when expanding the electronic functionality inside the car. Determining reasonable bounds for delays of safety critical data transmission depends not only on the performance and access mechanism of the communication system, but also requires reliable data transfer with respect to transmission errors and retransmissions, robustness of the bus system and control units, e.g. in terms of electromagnetic compatibility, and appropriate concepts for safe and efficient processing of the transmitted data. Especially safety critical functions, e.g. ESP (Electronic Stability Program) or ACC (Adaptive Cruise Control) demand reliable, robust and high-speed data transmission in order to work efficiently. With

the properties as described in detail in section 3, CAN [1] offers a sustainable performance to fulfill these demands, mainly regarding the available data rate and collision avoidance at media access.

The majority of car manufacturers employ CAN based data buses, however, there are several other competing bus systems, often coexisting with CAN inside a car, e.g. LIN (Local Interconnect Network) [2], MOST (Media Oriented Systems Transport) [3] or FlexRay [4]. They differ in data rate, media access and multiplexing schemes and other, mostly hardware related peculiarities. Analyzing data traffic with respect to bus specific parameters, network topology and communication paradigms is a necessary task, as the results of the investigations support the design of a reliable and robust communication system by adjusting the traffic settings appropriately.

The objective of this paper is to show the applicability of the method of Network Calculus for the evaluation of automotive communication systems. In such systems the mean values for performance measures, as obtained from stochastic modeling approaches like Queuing Theory, are of minor interest, as they are not sufficient to predict whether hard real-time deadlines are met in any case. Reliable operation and avoidance of system malfunctions with catastrophic effects mainly depend on the worst case performance of the communication infrastructure. The application of a deterministic modeling technique like Network Calculus yields upper delay bounds for data transmission which are inevitable to assess the reliability of the system in all possible scenarios of operation, since the *timeliness* itself is an essential aspect of hard real-time systems.

The paper is organized as follows: Section 2 presents previous approaches to analyze timing aspects of CAN communication and in section 3, the main features of CAN are presented briefly. Section 4 introduces the basic idea of Network Calculus, how actual input data can be used to construct the modeling elements *arrival* and *service curve* and how to determine delay bounds from these. Section 5 applies the findings from the previous section to both a short exemplary data set and real-world CAN bus traffic data. Finally, section 6 summarizes the results and gives an outlook for future work on this topic.

2 Related Work

CAN has been the object for various studies concerning the timing aspects of data transmission. [5] uses *Response Time Analysis* to determine whether deadlines of tasks are met for a given schedule. In [6] and [7], the scope of this analytic technique is extended towards an improved priority scheduling policy and automatic assignment of task and message periods. [8] investigates CAN using a system representation as timed automata and [9] presents a tool-supported exploration and optimization of a CAN bus design space. [10] applies the *Earliest Deadline First* algorithm in conjunction with the CAN protocol for an optimized assignment of task and message cycles. The drawback of all these proposals is that a static a-priori schedule for all tasks and messages is inevitable to analyze the real-time properties and to optimize the traffic with respect to timing de-

mands. In fact, due to the asynchronous wake-up of controllers along a CAN bus, such a global schedule can never be anticipated. To overcome this, the evaluation procedure that we propose requires only the statically assigned CAN identifiers and the specific cycle times at which each message is sent as input data. We do not need to know the global bus-wide schedule of traffic to obtain reasonable upper delay bounds for each priority class.

3 The CAN Bus

CAN development has been started in the 1980s by Robert Bosch GmbH, aiming to design a bus system for the specific needs of automotive applications. The resulting CAN bus is now standardized as ISO 11898 [1]. It uses a differential serial line architecture with *dominant* and *recessive* bits where a dominant bit represents a logical 0 and a recessive bit a logical 1. An idle bus has recessive level. Due to the open collector logic, if one station on the bus sends a dominant bit while another controller sends a recessive bit, the dominant bit wins, i.e. the bus is considered as logical 0. This feature allows to use a bit-wise arbitration scheme for the medium access, often called *CSMA/BA* (Carrier Sense Multiple Access with Bitwise Arbitration). Using this mechanism, each station listens to the bus while sending data. If a collision occurs where one station tries to send a recessive bit but receives a dominant bit, it will notice that another station is sending simultaneously and will stop its own transmission immediately. This makes the arbitration non-destructive, since the station sending the dominant bit can continue to send without any negative effects on the bus while the station sending the recessive bit remains silent from the time on where the collision has occurred. A retransmission of the interrupted frame is triggered automatically. Before sending, each controller listens to the bus and starts sending only if the bus has been recessive for at least 6 bit times (CS phase). A *NRZ* (Non-Return-to-Zero) encoding is used for the line encoding of the bits, where the sender inserts a complementary stuff bit when no change in the logic level has occurred over five successive bits. These stuff bits are removed automatically by the receiver of the message. This mechanism provides a base for synchronization and assures that a potential sender receives at least one dominant bit during the six bit times carrier sense phase if another station is sending.

CAN does not employ explicit sender or receiver addresses, but uses unique *message identifiers* to describe the content of the respective CAN frame. The frames are broadcast on the bus and each station can decide if the message content is relevant by examining the message identifier of received frames. These identifiers have to be assigned statically during the design phase of the bus system to avoid ambiguity in the interpretation of the frame content. A global view of the complete system is needed in this process.

There are two variants of CAN frames: standard frames with 11 bit message identifiers and extended frames with 29 bit identifiers. Both can coexist on the same bus. The payload can be of variable size, but for our application, all frames are standard CAN frames with 11 bit message identifiers and always contain 64

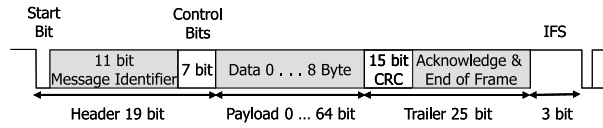


Fig. 1. CAN Frame Structure

bits of payload. Thus, when considering a maximum number of 19 stuff bits and including 3 bit times of inter-frame space (IFS), the maximum frame size is 130 bits.

The transmission of a CAN frame starts with one dominant start bit immediately followed by the message identifier from the most significant to the least significant bit. The structure of a standard CAN frame is shown in figure 1. Due to the media access scheme described above, the message identifiers create an implicit hierarchy of priorities. If more senders start to send simultaneously, the sender transmitting the frame with the highest message identifier has to send a recessive 1 bit that is overwritten by a dominant 0 bit first due to the binary encoding of the message identifiers in the frame header. While listening to the bus during the send process, it will notice the collision (dominant bus while sending a recessive bit) and stop sending. This process will continue until only one sender remains sending. This is always the one with the lowest message identifier and thus with the highest priority [11, 12].

4 Network Calculus

Network Calculus is a system theory for deterministic queuing systems, based on min-plus algebra. It plays a role similar to classical system theory for the analysis of electronic circuits, where in Network Calculus addition is replaced by computation of the infimum and multiplication becomes addition. Its main focus is on determination of bounds on worst case performance. One aim is to determine lower and upper bounds for end-to-end delays of nodes or collection of nodes within a network, for traffic backlog and for output limitations. By means of these performance-analytic bounding values – characterizing worst-case behavior of traffic flows – it is possible to dimension the corresponding buffers or to limit bursts. In the following section we will only give a short introduction to Network Calculus. A comprehensive overview can be found in [13], [14] or [15].

4.1 Theoretical Foundations

The most important modeling elements are the *arrival* and the *service curves* together with the *min-plus convolution*.

Let F be a flow (bits, messages, packets, etc.) into a system S and let $x(t)$ be the amount of data of F arriving in the time interval $[0, t]$ and $y(t)$ the amount of data leaving S in the time interval $[0, t]$. $x(t)$ is the arrival function of flow F .

By definition, $x(0)$ is zero, and $x(t) \geq x(s)$ for all $t \geq s$. The *delay at time t* for a lossless system is described as follows: $d(t) = \inf\{\tau : x(t) \leq y(t + \tau)\}$.

An upper bound on the arrival function $x(t)$ can be defined by the so-called arrival curve $\alpha(t)$:

Definition 1 (Arrival Curve) Let $\alpha(t)$ be a non-negative, non-decreasing function. Flow F is constrained by or has arrival curve $\alpha(t)$ iff $x(t) - x(s) \leq \alpha(t - s)$ for all $t \geq s \geq 0$.

Example 1 A commonly used arrival curve is the token bucket constraint:

$$\alpha_{r,b}(t) = b + rt \text{ for } t > 0 \text{ and zero otherwise.}$$

As one can see in figure 2, this arrival curve forms an upper limit for input $x(t)$ with (average) rate r and instantaneous burst b . That means $x(t) - x(s) \leq \alpha_{r,b}(t - s) = b + r(t - s)$. For $\Delta t := t - s$ and $\Delta t \rightarrow 0$ it holds that

$$\lim_{t \rightarrow s} \{x(t) - x(s)\} \leq \lim_{\Delta t \rightarrow 0} \{r \cdot \Delta t + b\} = b.$$

An important definition in Network Calculus is the following one:

Definition 2 (Min-plus Convolution) Let $f(t)$ and $g(t)$ be non-negative, non-decreasing functions that are 0 for $t \leq 0$. A third function, called *min-plus convolution*, is defined by

$$(f \otimes g)(t) = \inf_{0 \leq s \leq t} \{f(s) + g(t - s)\}.$$

Applying definition 2 we can characterize the arrival curve $\alpha(t)$ with respect to $x(t)$ as

$$x(t) \leq (x \otimes \alpha)(t).$$

The concept of arrival curves describes an upper bound to an input stream of a system processing some type of data. Concerning the output of this system, we are interested in service guarantees, i.e. is there a guaranteed minimum output $y(t)$ – the amount of data leaving system S ? The modeling element *service curve* deals with this question.

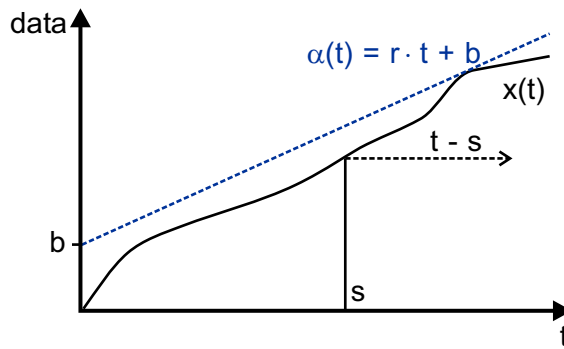


Fig. 2. Token Bucket Arrival Curve

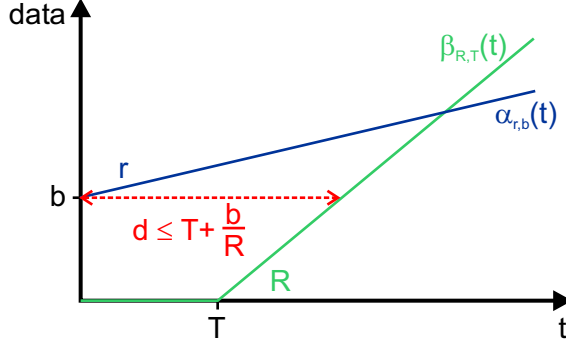


Fig. 3. Example for the Delay Bound

Definition 3 (Service Curve) Given is a system S with input flow $x(t)$ and output flow $y(t)$. The system offers a (minimum) service curve $\beta(t)$ to the flow iff $\beta(t)$ is a non-negative, non-decreasing function with $\beta(0) = 0$ and $y(t)$ is lower bounded by the convolution of $x(t)$ and $\beta(t)$:

$$y(t) \geq (x \otimes \beta)(t).$$

Example 2 One commonly used service curve is the rate-latency function: $\beta(t) = \beta_{R,T}(t) = R \cdot [t - T]^+ := R \cdot \max\{0; t - T\}$. The rate-latency function reflects a service element which offers a minimum service of rate R after a worst-case latency of T .

In figure 3, the graph $\beta_{R,T}(t)$ is a rate-latency service curve with rate R and latency T .

Theorem 1 (Delay Bound) Assume a flow constrained by arrival curve $\alpha(t)$ passing a system with service curve $\beta(t)$. The maximum delay d is given as the supremum of all possible delays of data, i.e. is defined as the supremum of the horizontal deviation between the arrival curve and the service curve:

$$d \leq \sup_{s \geq 0} \{ \inf \{ \tau : \alpha(s) \leq \beta(s + \tau) \} \}.$$

Example 3 Suppose there is a system with input according to a token bucket, thus $x(t) - x(s) \leq \alpha_{r,b}(t - s)$, and rate-latency output:

$$y(t) \geq \inf_{s \leq t} \{ x(s) + \beta_{R,T}(t - s) \}, \quad r \leq R.$$

Based on the theorems above, we determine the delay bound $d \leq b/R + T$, figure 3 shows the result.

4.2 Input Data Basis

For investigations of real-life CAN communication, the German car manufacturer Audi provided us with data for drivetrain CAN bus segments of several up-to-date model series (cf. section 5).

A common way of providing information about actual CAN data traffic in automotive application areas is to maintain a data set (often referred to as “Communication Matrix”) of message parameters and sender/receiver dependencies. As shown later, knowing the CAN Identifiers (CAN ID) and the cycle times of the periodically sent CAN messages is sufficient to apply Network Calculus for determination of worst case delays. In the remaining part of the paper the distinction between “CAN ID” and “priority class” (or simply “priority”) is made. Ordering the messages in the data set from lowest to highest ID in use, we abstract from the actual identifier and introduce a ranking of priorities where class 0 corresponds to the lowest ID and hence has highest priority, class 1 complies with the next-lowest ID, class 2 with the third-lowest ID and so on.

4.3 Generation of Arrival Curves

A central issue of performance evaluation with Network Calculus is the generation of appropriate arrival curves for data traffic. As explained before, the actual CAN media access is performed cyclic, successful transmission depends on the priority of the message to be sent compared to the priority of data which is transferred via the bus simultaneously.

Let N denote the number of the lowest priority class and $n \in \{0, 1, \dots, N\}$ be a priority for one particular CAN frame. Basically, there are three types of priorities: The “own” priority n (a certain message belongs to), the “higher” priorities 0 to $n - 1$ (which win the competitive media access) and the “lower” priorities $n + 1$ to N (which this priority n message dominates at media access, hence we can neglect them in the following). Since the arrivals of higher priority data affect the transmission of lower priorities, we need to consider them explicitly by forming an individual arrival curve (cf. section 4.4). It represents the cumulative arrivals of messages during a specified time interval at which no media access is possible for class n due to higher prioritized traffic. The highest priority class 0 has to be treated separately, since here no arrival curve for higher priorities exists and hence the arrival curve for higher classes is constantly zero.

To clarify the idea, the generation of arrival curves for the actual CAN priority classes is explained in detail in the following. It is based on the assumption that at each discrete point of time, all messages with matching cycle period occur at once. Hence, the lower the priority of the message, the more higher prioritized data may access the bus at the same time, prolongating the time until the media access for this particular message will be successful. This approach is valid, since performance evaluation with Network Calculus aims at determination of upper (worst case) delay bounds.

The data provided by Audi shows that a message of priority n will be sent in cycles of length c_n . Without loss of generality, c_n are integer values in the

following, depending on application specific demands for update rates of information exchange between CAN ECUs. A new message of priority n will be sent immediately after times $k \cdot c_n, k \in \mathbb{N}_0$, i.e. at integer multiples of c_n . Let C denote the least common multiple (lcm) of all c_n . At this point of time, messages of all priorities are sent, and since all messages are sent in a cyclic manner, the cycle of message arrivals will repeat in the same sequence for all messages of all priorities after the total cycle time $C = \text{lcm}\{c_0, c_1, \dots, c_N\}$.

The arrival curve for a message of class n can be written as the following *step function*:

$$\alpha_n(t) = \left\lceil \frac{t}{c_n} \right\rceil \cdot l,$$

where $l = 136$ bits denotes the maximum frame length including the 6 bit CS time.

The cumulative number of arrivals of messages with *priority higher than* n can be calculated as

$$\hat{\alpha}_n(t) = \sum_{i=0}^{n-1} \alpha_i(t) = \sum_{i=0}^{n-1} \left\lceil \frac{t}{c_i} \right\rceil \cdot l.$$

A linear function $\bar{\alpha}_n(t)$ can be determined as an upper bound for the cumulative higher priority arrivals $\hat{\alpha}_n(t)$ as

$$\begin{aligned} \bar{\alpha}_n(t) &= n \cdot l + \frac{\sum_{i=0}^{n-1} C/c_i \cdot l}{C} \cdot t \\ &= n \cdot l + \sum_{i=0}^{n-1} \frac{l}{c_i} \cdot t \\ &= \sum_{i=0}^{n-1} \left(\frac{t}{c_i} + 1 \right) \cdot l. \end{aligned}$$

As we will show later, using $\bar{\alpha}_n(t)$ instead of $\hat{\alpha}_n(t)$ is a crucial factor for determining appropriate and definition compliant service curves for the single priority classes.

We construct the curve of $\bar{\alpha}_n(t)$ using the following idea: We define a token bucket arrival curve $\bar{\alpha}_n(t)$. Therefore, we have to come up with the vertical offset and the slope of the curve. The offset is obtained by adding all single offsets (i.e., we assume that all possible bursts happen initially immediately after time $t = 0$). The slope is obtained by adding the slopes bounding all single higher priority arrival curves. These slope boundaries are given by $\frac{C/c_i}{C} \cdot l = l/c_i$, because exactly C/c_i messages of length l with priority i are generated in each interval of length C . As one can easily see, $\bar{\alpha}_n(t) \geq \hat{\alpha}_n(t)$. Thus, $\bar{\alpha}_n(t)$ is an upper bound for the cumulative arrival of higher priority messages that fulfills all preconditions for a Network Calculus arrival curve. Defining $b_n := n \cdot l$ and $r_n := \sum_{i=0}^{n-1} (l/c_i)$ allows us to write this function in token bucket form

$$\bar{\alpha}_n(t) = b_n + r_n \cdot t.$$

4.4 Determination of Service Curves and Calculation of Delay Bounds

Now we will show how Network Calculus can be applied to compute the time critical maximum frame transfer delay of one bus controller station. After we determined a step function as an arrival curve for the input traffic in section 4.3, we have to look for a “good” service curve reflecting the behavior of a CAN node as far as possible. Like in many other worst case modeling situations, a rate-latency service curve $\beta_{R,T}(t)$ is adequate to model the outgoing data.

First of all, for a high-speed CAN the rate R is given by 500 kbps for all message frames, independent of their priority. Considering the inherent hierarchy of frame priorities and the non-preemptive scheduling mechanism of data sending the following holds: Assuming the worst case, a higher priority frame always has to wait until a frame, possibly of lower priority, is completely sent. Since the maximum frame size is 130 bits and the CS time is 6 bit times, the worst-case sending time of such lower priority data is $T = l/R = 136 \text{ bits}/500 \text{ kbps} = 0.272 \text{ ms}$.

Thus, for the rate-latency service curve $\beta_{R,T}(t)$ it holds:

$$\begin{aligned}\beta_{R,T}(t) &= R \cdot [t - T]^+ \\ &= 500 \text{ kbps} \cdot [t - 0.000272 \text{ s}]^+.\end{aligned}$$

The next problem we have to solve deals with the priority scheduling of data frames. We compute the guaranteed worst case delays of all this traffic of different priority based on theorem 1. We use the arrival and service curve only and propose the following approach:

Let $x_j(t)$ denote the input data and $y_j(t)$ the output data at time t per controller of priority $j \in \{0, \dots, N\}$. The procedure starts with input $x_0(t)$ and output $y_0(t)$. Theorem 1 tells us that delay

$$d_0 \leq \sup_{t \geq 0} \{\inf\{\tau : \alpha_0(t) \leq \beta(t + \tau)\}\}$$

with $\alpha_0(t) = \text{step-function-type arrival curve of input } x_0(t)$ and $\beta(t) = \beta_{R,T}(t) = 500 \text{ kbps} \cdot [t - 0.000272 \text{ s}]^+$. Then we have to determine the maximum delay d_1 for the next lower priority frames of the same controller station. However, it would not be correct to take the same service curve $\beta(t)$, because the frames of priority 1 will be served only after sending frames of priority 0 is finished. We follow the approach of aggregate traffic modeling as presented in [16] and also used for CPU task scheduling with preemption in [17] and construct the service curve

$$\beta_1(t) := [\beta(t) - \bar{\alpha}_1(t)]^+.$$

Even though $\hat{\alpha}_1(t) = \alpha_0(t)$ is a valid arrival curve, constructing a service curve according to $\beta_1(t) := [\beta(t) - \hat{\alpha}_1(t)]^+$ yields no valid cumulative service curve, as shown in figure 4. Thus, we need to use $\bar{\alpha}_1(t)$ instead of $\hat{\alpha}_1(t)$ to guarantee that $\beta_1(t)$ is a non-decreasing function.

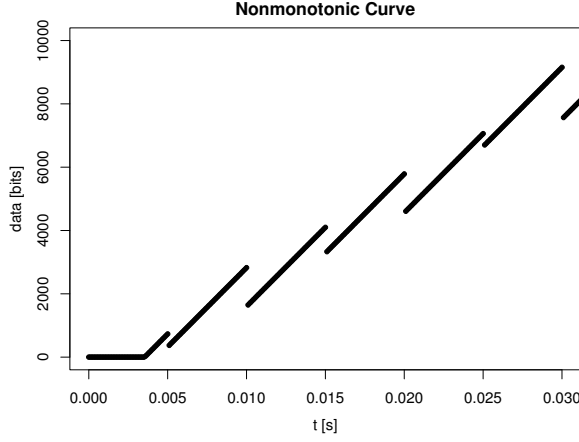


Fig. 4. Nonmonotonic Curve

The maximum delay d_1 for frames of priority 1 is

$$d_1 \leq \sup_{t \geq 0} \{ \inf \{ \tau : \alpha_1(t) \leq \beta(t + \tau) - \bar{\alpha}_1(t + \tau) \} \}.$$

Going on in this way from priority j to the next lower one $j + 1$, we must build a new service curve by diminishing the previous one by the arrival curve $\bar{\alpha}_{j+1}(t)$ for all frames with priority $0, 1, \dots, j$. So, at each priority step only two sorts of frames are considered: the frames of present priority $x_j(t)$ and the sum of all higher priority frames $\hat{x}_j(t) = \sum_{i=0}^{j-1} x_i(t)$. All frames of lower priority ($j + 1, \dots, N$) have no impact besides maybe a single frame just in service which has already been modeled by the “non-preemptive” scheduling in the latency component T of $\beta_{R,T}(t)$.

The following procedure summarizes our approach:

$\forall j \in \{0, 1, \dots, N\}$ and $t \in [0, \infty]$:

$$\begin{aligned} \alpha_j(t) &= \left\lceil \frac{t}{C_j} \right\rceil \cdot l \\ \bar{\alpha}_j(t) &= b_j + r_j \cdot t \\ \beta_j(t) &= [\beta(t) - \bar{\alpha}_j(t)]^+ \\ &= [R[t - T]^+ - (b_j + r_j \cdot t)]^+ \\ &= [R(t - T) - (b_j + r_j \cdot t)]^+ \\ &= \left[(R - r_j) \cdot \left(t - \frac{RT + b_j}{R - r_j} \right) \right]^+. \end{aligned}$$

Again, we obtain a rate latency service curve with rate $R'_j := R - r_j$ and latency $T'_j := (RT + b_j)/(R - r_j)$.

Lastly, this yields

$$d_j \leq \sup_{t \geq 0} \{ \inf \{ \tau \geq 0 : \alpha_j(t) \leq \beta_j(t + \tau) \} \}.$$

Due to the actual shape of the arrival curve $\alpha_j(t)$ and the service curve $\beta_j(t)$ in our case, this inequation can be solved geometrically by calculating the intersection point of $\beta_j(t)$ with the height of the first step of the arrival curve $\alpha_j(t)$, which is exactly $l = 136$ bits. This yields the following results:

$$\begin{aligned} d_j &\leq \frac{l}{R'_j} + T'_j \\ &= \frac{l + RT + b_j}{R - r_j} \\ &= \frac{(j + 2) \cdot l}{R - \sum_{i=0}^{j-1} (l/c_i)}. \end{aligned}$$

5 Application

5.1 Exemplary Message Schedule

To illustrate the process of generating the arrival curves and calculating the service curves and delays, assume that only five different CAN priorities with respective arbitrarily chosen cycle times as shown in table 1 are used and that all messages are standard CAN frames with a maximum length of 130 bits, therefore $l = 136$ bits. The results obtained for a CAN data rate of 500 kbps are also shown in table 1.

		$\bar{\alpha}_i(t)$		$\beta_i(t)$		
i	c_i	b_i	r_i	R'_i	T'_i	d_i
	[ms]	[bits]	[bps]	[bps]	[ms]	[ms]
0	50	0	0	500000	0.272	0.544
1	10	136	2720	497280	0.547	0.820
2	100	272	16320	483680	0.844	1.125
3	20	408	17680	482320	1.128	1.410
4	30	544	24480	475520	1.430	1.716

Table 1. Example Results

Figure 5 illustrates these results. The arrival curve $\alpha_i(t)$ for each priority i is shown in the first column of this figure. In the second column the staircase function $\hat{\alpha}_i(t)$ is depicted in black steps and the linear function $\bar{\alpha}_i(t)$ is drawn as a solid gray line. The third column shows the arrival curve $\alpha_i(t)$ as a solid horizontal line and the corresponding service curve $\beta_i(t)$ as a dashed line. The

horizontal distance between the thin vertical lines marks the geometrically determined maximum delay d_i as the intersection point of $\beta_i(t)$ and $l = 136$ bits. Please note the use of different scales for both the time and the data axis in the third column compared to the first two columns in this figure. Otherwise, the intersection point would not be visible due to the different orders of magnitude in temporal and data dimension of the arrival and service curve.

5.2 Actual CAN Traffic

To emphasize the relevance and applicability of our approach, Audi supported our investigations with CAN data sets for various current series. We analyzed data for three different models, in the following referred to as model A, B and C. The cycle times for the model dependent varying number of active CAN priority classes are depicted in figure 6.

Two priority classes are highlighted in white and black in the graphics of figures 6 and 7. They represent messages which are semantically identical, i.e. contain the same information in each model. The priority class they are assigned to and the number of higher priority classes present differ from one model to the other. Hence, we expect different worst case delays for the same content in varying traffic load and scheduling scenarios. To determine actual values for all three models, we applied our method to the corresponding data sets. Figure 7 shows the maximum delays calculated for model A over both the CAN ID and priority class.

In figure 8 the worst case delays are depicted explicitly for the two messages we selected for comparison beforehand. The values show significant deviations from one model to another and thus a strong influence of the priority class of the actual message and the higher prioritized traffic on the maximum delay at CAN media access.

6 Conclusions and Future Work

We have shown in this paper that application of Network Calculus yields reasonable results for deterministic performance evaluation of state-of-the-art automotive communication networks. The delay bounds we obtained support the decision whether for CAN frames of arbitrary priority hard real-time transmission properties can be fulfilled by a given traffic schedule in any constellation. Our approach is straightforward and easily applicable to actual CAN traffic, providing a sound analytical method to determine delay bounds for all CAN priority classes.

Thus, for future system layouts, more sophisticated network evaluation with respect to dependability and functionality, especially for safety critical applications, is possible. Further research on this topic will concentrate on the modeling of arrival and service curves. The question is whether refined constructs for determination of upper-bound thresholds can be found, resembling the actual data traffic even closer, yet compliant to the definitions of modeling elements according to Network Calculus.

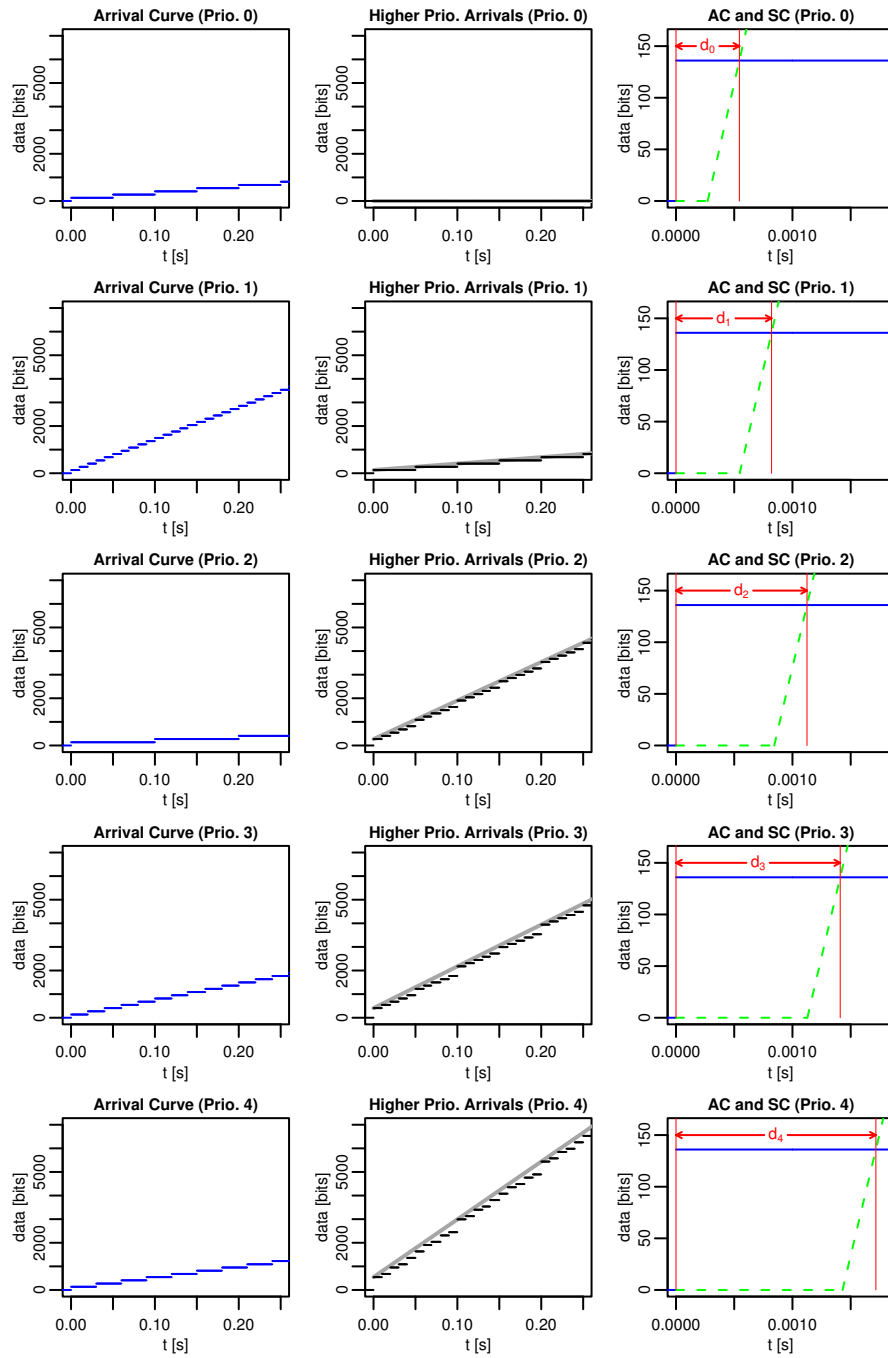


Fig. 5. Analysis Results for the Example

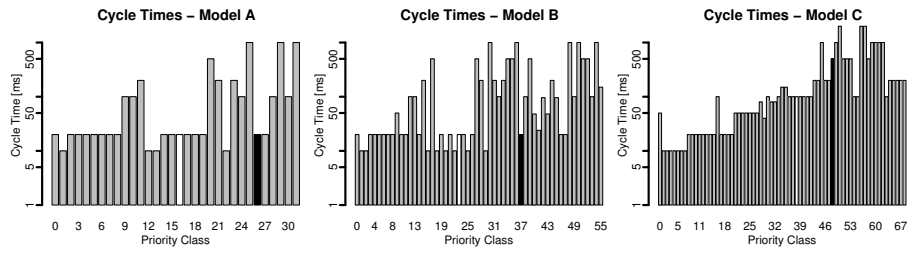


Fig. 6. Cycle Times of Drivetrain CAN Traffic - Model A, B and C

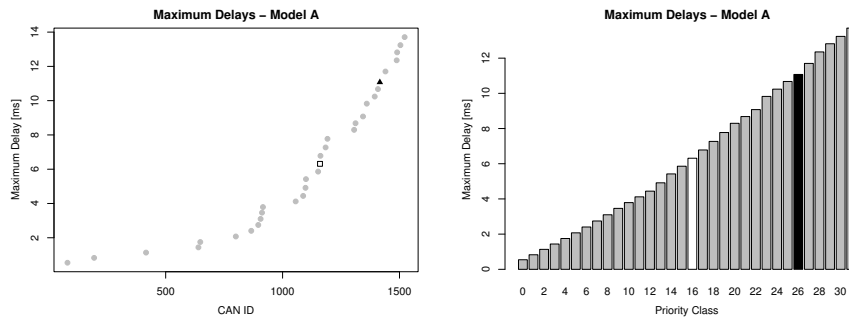


Fig. 7. Worst Case Delays Ordered by CAN ID (left) and Priority (right) - Model A

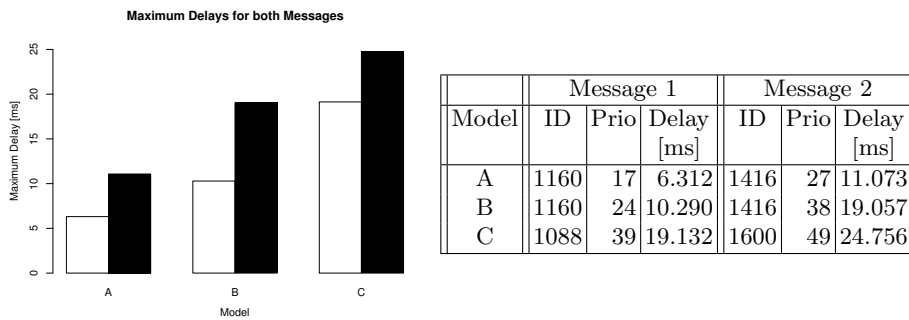


Fig. 8. Comparison of Worst Case Delays in Model A, B and C

References

1. International Standard ISO 11898: Road vehicles – Interchange of Digital Information – Controller Area Network (CAN) for High Speed Communication. first edn. International Organization for Standardization (1994) ISO Reference Number ISO 11898:1993(E).
2. v. d. Wense, H.C., ed.: LIN Specification Package. LIN Consortium (2003)
3. MOST Cooperation: MOST Media Oriented Systems Transport. Rev 2.4 edn. (2005)
4. FlexRay Consortium: FlexRay Communications System Protocol Specification. Version 2.1 edn. (2005)
5. Tindell, K., Burns, A.: Guaranteed Message Latencies for Distributed Safety Critical Hard Real-Time Networks. Technical Report YCS 229, Dept. Computer Science, Univeristy of York (1994)
6. Davis, R.I., Burns, A., Bril, R.J., Lukkien, J.J.: Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised. *Real-Time Syst.* **35** (2007) 239–272
7. Davare, A., DiNatale, M., Zhu, Q.: Period Optimization for Hard Real-time Distributed Automotive Systems. In: Proceedings of the 44th IEEE/ACM Design Automation Conference. (2007)
8. Krakora, J., Hanzalek, Z.: Verifying Real-Time Properties of CAN bus by Timed Automata. In: In FISITA 2004 - World Automotive Congress, Barcelona. (2004)
9. Hamann, A., Racu, R., Ernst, R.: Formal Methods for Automotive Platform Analysis and Optimization. In: In Proc. Future Trends in Automotive Electronics and Tool Integration Workshop (DATE Conference), Munich. (2006)
10. Richardson, P., Sieh, L., Elkateeb, A., Haniak, P.: Real-time Controller Area Networks (CAN) – managing transient surges. *Integr. Comput.-Aided Eng.* **9** (2002) 149–165
11. Lawrenz, W., ed.: CAN Controller Area Network. fourth edn. Hüthig Verlag, Heidelberg (2000)
12. Etschberger, K.: Controller-Area-Network. second edn. Carl Hanser Verlag, München (2000)
13. Le Boudec, J.Y., Thiran, P.: Network Calculus. Springer Verlag LNCS 2050 (2001)
14. Cruz, R.L.: A calculus for network delay, Part I: Network elements in isolation. *IEEE Transactions on Information Theory* **37** (1991) 114–131
15. Cruz, R.L.: A calculus for network delay, Part II: Network analysis. *IEEE Transactions on Information Theory* **37** (1991) 132–141
16. Fidler, M., Sander, V.: A parameter based admission control for differentiated services networks. *Computer Networks* **44** (2004) 463–479
17. Wandeler, E., Thiele, L.: Real-time interfaces for interface-based design of real-time systems with fixed priority scheduling. In: EMSOFT '05: Proceedings of the 5th ACM international conference on Embedded software, New York, NY, USA, ACM Press (2005) 80–89