Stochastic Analysis of CAN-Based Real-Time Automotive Systems

Haibo Zeng, Marco Di Natale, *Member, IEEE*, Paolo Giusto, and Alberto Sangiovanni-Vincentelli, *Fellow, IEEE*

***Abstract*—Many automotive applications, including most of those developed for active safety and chassis systems, must comply with hard real-time deadlines, and are also sensitive to the average latency of the end-to-end computations from sensors to actuators. A characterization of the timing behavior of functions is used to estimate the quality of an architecture configuration in the early stages of architecture selection. In this paper, we extend previous work on stochastic analysis of response times for software tasks to controller area network messages, then compose them with sampling delays to compute probability distributions of end-to-end latencies. We present the results of the analysis on a realistic complex distributed automotive system. The distributions predicted by our method are very close to the probability of latency values measured on a simulated system. However, the faster computation time of the stochastic analysis is much better suited to the architecture exploration process, allowing a much larger number of configurations to be analyzed and evaluated.**

***Index Terms*—Controller area network (CAN), distributed systems, stochastic analysis.**

I. Introduction

THE complexity of automotive electronic systems is rapidly growing. A modern vehicle contains between 20 and 100 electronic control units (ECUs), with several million lines of embedded software code, networked over standard communication buses. Controller area networks (CANs, up to 10) are the majority of the links. Furthermore, many automotive electronic systems involve real-time control of mechanical parts, such as chassis, powertrain, and active safety.

Car electronic architectures need to be defined, evaluated, and selected years in advance, when the functions they will support are only partly defined. This stage is of enormous importance for its implications on cost, performance and extensibility. Currently, it is driven by a *what-if* iterative process. First, a set of metrics and constraints is defined; then, a few candidate architectures are evaluated based on the analysis results, and a solution is selected.

In this work, we focus on the stochastic analysis of the timing performance of distributed automotive architecture with priority-based scheduling, i.e., OSEK compliant operating systems [3] and the CAN bus protocol [2]. The communication model considered in this research is the *periodic activation* model, where all tasks are activated periodically, and communicate by means of asynchronous buffers preserving the latest value written on them. This model is supported by AUTOSAR, the open standard for the description of automotive software architectures [1]. Message transmission is triggered periodically and each message contains the latest values of the signals that are mapped into it. At each stage, the consumer of the data may need to wait up to an entire period (*sampling delay*) to get the latest data in the buffer.

A. Research Motivation

The work presented in this paper is motivated by the study of current and future *active safety functions*. These functions gather a 360° view of the environment via radars and cameras, and require several processing stages before the actuation signals are produced, including sensor fusion, object detection, control and arbitration layers. Examples are Adaptive Cruise Control (ACC) or Lane Keeping Systems. In an active cruise control system, a set of radars (or other sensors) scans the road in front of the car to ensure that there are no other cars or objects moving at a lower speed or even stopped in the middle of the lane. If such an object is detected, the system lowers the target speed of the cruise control until it matches the speed of the detected obstacle.

A hard deadline can be defined for such a system (the worst case reaction time that allows preventing a collision), but clearly a faster average reaction time is always preferable and the designer can leverage additional knowledge on the probability distribution of the function response times.

End-to-end latencies for computations that span over several ECUs and CAN buses are a function of task response times, message response times, and communication delays. If the communication model is by periodic sampling, then the latter depends on the relative phases of task and messages.

Worst case analysis based on schedulability theory allows to compute the contribution of tasks and messages [6] to end-to-end latencies and provides the architecture designer with a set of values (one for each end-to-end path) on which he/she can check correctness of an architecture solution. However, worst case analysis should be complemented by probabilistic analysis for two main reasons.

- Many applications are not time-critical, but the performance of the controls depend on the average response time, which needs to be analyzed and minimized. This is true also for many time-critical functions.
- In the periodic activation model, each time a message is transmitted or received, a task (message) may need to wait

Manuscript received December 14, 2008; revised April 20, 2009 and August 14, 2009. First published September 29, 2009; current version published November 06, 2009. Paper no. TII-08-12-0213.R2.

H. Zeng and P. Giusto are with General Motors R&D, Palo Alto, CA 94360 USA (e-mail: haibo.zeng@gm.com; paolo.giusto@gm.com).

M. Di Natale is with Scuola Superiore S. Anna, Via Moruzzi, 1, 56127 Pisa, Italy (e-mail: marco@sssup.it).

A. Sangiovanni-Vincentelli is with the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA 94720 USA (e-mail: alberto@eecs.berkeley.edu).

Digital Object Identifier 10.1109/TII.2009.2032067

up to an entire period of *sampling delay* to read (forward) the latest data stored in the communication buffers. Adding worst case delays at each step allows to obtain the worst-case latencies of paths, but the probability of such values can be very small. So small in fact (in the path we considered it is less than 10^{-14} with a period of 50 *ms*, i.e., $7.2 \times 10^{-10}/\text{hour}$), to be smaller than the probability of HW failure (e.g., ISO 26262 ASIL level D with targeted random hardware failure $10^{-9} \sim 10^{-8}/\text{hour}$ [20])! In this case, designing for worst-case time behavior can be wasteful.

B. State-of-the-Art

Probabilistic analysis of priority-scheduled systems has been addressed in the past. Gardner [8] computes the probability of deadline misses for each job released in the busy interval, and chooses the maximum as an upper bound on the probability of deadline misses for the corresponding task. Simulation-based analysis and stochastic methods exist [7], [13] for computing the probability density functions (pdfs, or probability mass functions-pmfs-in the discrete case) of the response times of tasks scheduled on single or multiprocessor platforms. For messages scheduled on a CAN bus, Navet *et al.* [15] introduce the concept of worst case deadline failure probability (WCDFP) because of transmission errors. WCDFPs are computed with respect to the *critical instant*, i.e., the worst case response time scenario. In [16], Nolte *et al.* extend the worst case response time analysis using random message transmission times that take into account the probability of a given number of stuff bits, but the analysis is still performed in the worst case scenario. Worst case analysis of task and message response times in priority-based systems addressed independent periodic tasks [11], tasks with offsets and jitter [18], and OSEK systems [12]. In [6], Davis *et al.* discuss scheduling and response time analysis of CAN messages.

In Lehoczky's Real-Time Queueing Theory [10], the task set is modeled as a Markovian process. Given a scheduling algorithm and the distribution of the deadlines, the distributions of the times remaining until the deadline of jobs are computed. The analysis is performed in heavy traffic conditions, when the behavior can be approximated by a diffusion process. Arrival and execution times are modeled by Poisson distributions, and the system has a very large set of tasks with very high utilization ($U \rightarrow 1$). In an independent but related work, Kim and Shin [9] model an application as a queueing network scheduled as first-in–first-out (FIFO), and use exponentially distributed task execution times. The underlying mathematical model is a continuous-time Markov chain.

Manolache presents an analytical method to compute the expected deadline-miss ratio for single processor systems, and an approximate method for multiprocessor systems [14].

The most relevant method to our approach was proposed by Díaz *et al.* [7], [13], in which the *pmf*s of the response times of a set of independent periodic tasks executed on a single processor by a preemptive priority-based scheduler is computed. The activation times of all the task instances are known and the task execution times are defined by *pmf*s. A summary of the results in [7], [13] is presented in Section IV.

C. Summary of the Paper

We provide the theory for the probabilistic analysis of the latency in the end-to-end propagation of information among periodically activated tasks and messages. A communication mechanism based on the preservation of the latest written value and the overwriting of old ones (shared variable buffer) is assumed. We target current automotive standards. Priority-based scheduling as in OSEK is assumed for ECUs, and messages are exchanged on CAN buses where they are transmitted in order of their IDs. Our experimental results show that our technique provides a good approximation of the latency distribution. We provide results from simulations and trace data, and we compare them with the results obtained by our stochastic framework.

In the following, we first describe the typical architecture of automotive distributed systems in Section II; then, in Section III, we define a formal model for it and we formalize the concept of end-to-end latency. The following three sections present the stochastic analysis framework for each of the components: In Section IV, we briefly summarize the previous work on periodic preemptable tasks. In Section V, the stochastic analysis framework that approximates CAN message response times is presented and evaluated by simulation and trace data. In Section VI, the stochastic analysis of end-to-end latency is addressed and evaluated. Finally, we provide the conclusions.

II. Distributed Automotive Architectures

A typical automotive system consists of several ECUs communicating on CAN buses. Software stacks provide support for the computations and communication, including application tasks, the middleware, drivers, and bus peripherals.

Application Tasks are the implementation of a control or dataflow algorithm. They are activated periodically as supported by the AUTOSAR standard [1], and scheduled according to their (static) priority by an AUTOSAR- or OSEK-compliant operating system [3]. For communication purposes, tasks read input signals at their activation and write their outputs in shared variables at the end. One special set of such shared variables is at the boundary between the application and the middleware.

The *Middleware* acts as the interface layer from the application to the CAN bus. The transmit task (*TxTask*) is a special middleware task that is activated periodically and has the responsibility of assembling each message from the signal values and then enqueueing it for transmission on the bus. Each message m_i has an associated period T_i, which is an integer multiple of the TxTask period. When a message is queued, the TxTask checks the CAN controller for empty transmit objects. In case it is found, the message frame is copied into it. Otherwise, it is copied into a priority sorted queue.

On the transmission side, the *CAN driver* is triggered by the interrupt signal that informs of the completion of a transmission. It selects the highest priority message in the transmit queue and places it into the empty transmit object.

The *CAN bus* is a wired AND channel connecting all nodes. Its scheduling policy is priority-based and *nonpreemptive* [2], that is, a message transmission can not be preempted by higher priority message instances. Messages have a maximum data payload of 64 bits and the frame format has a fixed size, with the only exception of the bit stuffing mechanism. The bus synchronization protocol does not allow more than five consecutive equal value bits. After five bits of the same value, an opposite bit is added by the transmitting node, and removed by the receivers before delivering the message.

III. System Model and Notation

The model for the distributed real-time system considered in this paper is the following: a periodic activation signal from a local clock triggers the computations on each ECU. Some application task reads the input data from a sensor, computes intermediate results that are sent over the network to other tasks and, finally, another task, executing on a remote node, generates the outputs as the result of the computation.

A natural model for the description of end-to-end computations is a *dataflow*. We restrict the dataflow to be acyclic, thus the system is a *Directed Acyclic Graph (DAG)* defined as a tuple $(\mathbf{V}, \mathbf{E})$, where $\mathbf{V}$ is the set of vertices representing the *tasks* and *messages* in the system, and $\mathbf{E}$ is the set of edges representing the *data signals* communicated among them.

$\mathbf{V} = \{o_1, \ldots, o_n\}$ is the set of objects implementing the computation and communication functions of the system, i.e., tasks and messages are considered equally for the purpose of the communication dataflow. For a task object, we also use the notation τ_i. A message object is denoted as m_i.

A task τ_i is characterized by $(\Upsilon_i, T_i, O_i, \mathcal{E}_i, P_i)$, where Υ_i is the ECU resource on which it is executed, T_i is its period, O_i its initial phase, $\mathcal{E}_i$ its execution time, and P_i its priority. $\mathcal{E}_i$ is a discrete random variable (calligraphic letters denote random variables) with a known distribution $f_{\mathcal{E}_i}$ defining a probability for each value within $[E_i^{\min}, E_i^{\max}]$.

We approximate continuous time with a *discrete-time model with granularity* τ. All probability distributions, including those of execution times, response times and backlogs (defined in Sections IV and V) are expressed as *pmf*s of time values that are multiples of the given tick unit τ.

For each periodic activation, we consider a task instance or *job*. We denote the jth job of τ_i as $\Gamma_{i,j}$, its *arrival or release time* as $A_{i,j} = O_i + (j-1)T_i$, which is the time $\Gamma_{i,j}$ is ready for execution. The start time of $\Gamma_{i,j}$ is $\mathcal{S}_{i,j}$, and its finish time is $\mathcal{F}_{i,j}$. We assume the execution times of all the jobs of τ_i are independent and identically distributed as $f_{\mathcal{E}_i}$, and are independent from the execution times of other tasks.

Each job $\Gamma_{i,j}$ reads its inputs at its release time $A_{i,j}$ and writes its results at the finish time $\mathcal{F}_{i,j}$. The response time $\mathcal{R}_{i,j}$ is defined as the time interval from $A_{i,j}$ to $\mathcal{F}_{i,j}$, which is a random variable. The task response time $\mathcal{R}_i$ is defined as the average of the response times $\mathcal{R}_{i,j}$ of its jobs.

A message object o_i, also denoted as m_i, is similarly defined by $(\Upsilon_i, \Upsilon_i^{\text{src}}, T_i, O_i, \mathcal{E}_i, P_i)$, where Υ_i is the bus resource needed for its transmission, and Υ_i^{src} is the ECU from which it is transmitted. For each periodic activation, we consider a *message instance*. We denote the j-th instance of message m_i as $M_{i,j}$, its arrival or queueing time as $A_{i,j} = Q_{i,j} = O_i + (j-1)T_i$, relative to the clock of Υ_i^{src}, its start time as $\mathcal{S}_{i,j}$, and its finish time as $\mathcal{F}_{i,j}$. Each message m_i is associated with a unique ID P_i, which also represents its priority. According to the CAN protocol, the lower is the message ID, the higher is its priority. Response times of messages and message instances are defined in a similar way as for tasks.

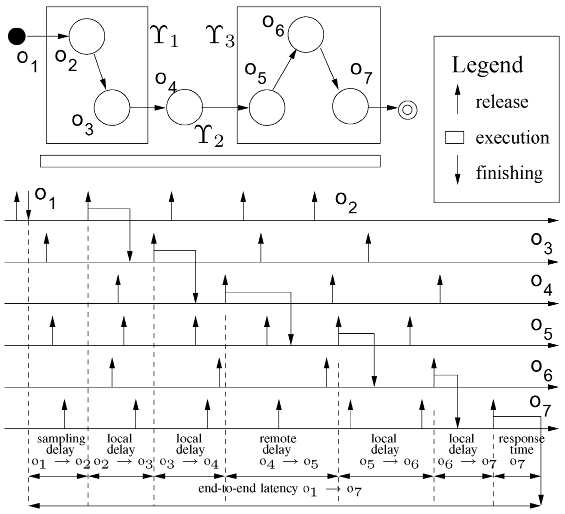

Υ_1
Υ_3
Υ_2
o_1
o_2
o_3
o_4
o_5
o_6
o_7
Legend
release
execution
finishing
o_1
o_2
o_3
o_4
o_5
o_6
o_7
sampling delay $o_1 \rightarrow o_2$
local delay $o_2 \rightarrow o_3$
local delay $o_3 \rightarrow o_4$
remote delay $o_4 \rightarrow o_5$
local delay $o_5 \rightarrow o_6$
local delay $o_6 \rightarrow o_7$
response time o_7
end-to-end latency $o_1 \rightarrow o_7$

Fig. 1. Model of an automotive architecture and its end-to-end latency.

For each resource Υ_k, we define its *hyperperiod* H_k as the least common multiple (lcm) of the periods of all objects executing on or transmitted on it, i.e., $H_k = \text{lcm}(\bigcup_i T_i)$, where $\Upsilon_i = \Upsilon_k$. The system-level hyperperiod H is the least common multiple of the resource hyperperiods.

We assume there is no clock synchronization among the nodes connected to the CAN bus. We define $\mathcal{O}_{\Upsilon_k,\Upsilon_l}$ as the clock difference between ECUs Υ_k and Υ_l, and we assume that all nodes boot up at arbitrary times. Hence, the *pmf* of $\mathcal{O}_{\Upsilon_k,\Upsilon_l}$ is uniformly distributed within the system hyperperiod. In the following, we need to express message queueing times relative to the clocks of remote ECUs. To this purpose, we need to define the random variable $\mathcal{O}_{\Upsilon_k,i}$ as the offset of m_i, a generic message queued on ECU_l, relative to the clock of a remote ECU_k. It is $\mathcal{O}_{\Upsilon_k,i} = O_i + \mathcal{O}_{\Upsilon_k,\Upsilon_l}$.

The edges $E = \{e_1, e_2 \ldots, e_m\}$ represent the data transfers between objects. An edge $e_i = (o_h, o_k)$ connects the output of o_h to the input of o_k. e_i carries a data signal produced by o_h and made immediately available to o_k. A *path* from o_i to o_j, denoted as $\Pi_{i,j}$, is an ordered sequence $(o_i, \ldots, o_j)$ of objects such that there is an edge between any two consecutive objects. Fig. 1 gives an example of a path between the objects o_1 and o_7, where the diagram on top left is the architecture of the resources and objects. The activation of the source object (o_1) represents the detection of an external event, and the completion of the sink object (o_7) represents the actuation output. Resources Υ_1 and Υ_3 are ECUs, Υ_2 is a CAN bus.

Each path consists of one or more local and remote interactions. In Fig. 1, local interactions are between tasks o_2 and o_3, and in the chain $o_5 \rightarrow o_6 \rightarrow o_7$. Each message is enqueued by a middleware task at the transmitting node. As such, the interaction between the transmitting task and the message is part of the local chain, like $o_3 \rightarrow o_4$ in the figure. Other interactions between a message and the receiving task, such as the edge (o_4, o_5), where the input and output objects are activated according to two unsynchronized clocks are considered

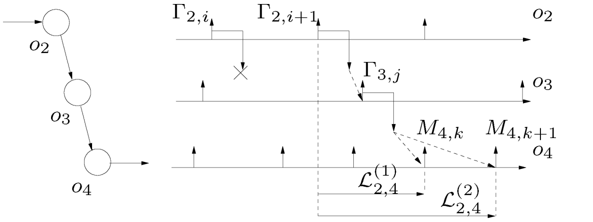


Fig. 2. Data loss and duplication during communication.

as *remote*. If the period of the reader o_k is larger than the period of o_h (under-sampling), some data values sent by o_h may be overwritten before they are read, and are never propagated. In the case of over-sampling ($T_k < T_h$), some data values are read by multiple instances of o_k. Consider the communication chain $o_2 \rightarrow o_3 \rightarrow o_4$ in Fig. 2. The data produced by the job $\Gamma_{2,i}$ of o_2 is never propagated because the next instance $\Gamma_{2,i+1}$ finishes execution and overwrites it before the instance $\Gamma_{3,j}$ of the receiving task is activated. On the other hand, the data generated by $\Gamma_{3,j}$ is read and propagated by two instances $M_{4,k}$ and $M_{4,k+1}$ of the following message o_4. For the path $\Pi_{i,j} = (o_i, \ldots, o_j)$, we are interested in the time interval required for an input at the source task o_i of the chain to be propagated to the last task o_j at the other end of the chain. If intermediate results are overwritten before they are read, the corresponding dataflows are not propagated and not considered in the analysis (like the datapath ending with the output of $\Gamma_{2,i}$ in Fig. 2). Also, if some data is read multiple times, we take into consideration all the latency values associated with this data item. In the figure, this happens for the two latency values $\mathcal{L}_{2,4}^{(1)}$ and $\mathcal{L}_{2,4}^{(2)}$. The *end-to-end latency* $\mathcal{L}_{i,j}$ of path $\Pi_{i,j}$ is defined as the interval between the activation of one instance of o_i and the completion of the instance of o_j that produces a result dependent on the output of o_i.

IV. Stochastic Analysis of Task Response Times

In [7] and [13], Díaz *et al.* present a stochastic analysis framework to compute the probability distribution of task response times for fixed priority systems where tasks are periodic, preemptable, and activated with known offsets (the activation times are determined). To calculate task response times, the *P-level backlog at time t* (denoted as $\mathcal{W}_t^P$) is defined as the sum of the remaining execution times of all the jobs that have priority higher than or equal to P and are not completed until t. The task release pattern in one hyperperiod is repeated for all the other hyperperiods, thus Díaz *et al.* focus on the P-level backlog observed at the beginning of each hyperperiod, denoted as $\mathcal{G}_k^P = \mathcal{W}_{(k-1)H}^P$. The stochastic process defined as the sequence of random variables $\{\mathcal{G}_1^P, \ldots, \mathcal{G}_k^P, \ldots\}$ is a Markov chain. In addition, a stationary distribution $\mathcal{G}^P$ of the P-level backlog $\mathcal{G}_k^P$ exists as long as (stability condition) the average system utilization is less than one. After the stationary distribution $\mathcal{G}^P$ is calculated, the stationary backlog at any time *within* the hyperperiod can be calculated by iteratively using two operations called *convolution* and *shrinking*. The P-level backlog *pmf* right after the release of a job with priority higher than or equal to P is obtained by performing a convolution, at the job release time, of the backlog *pmf* with the job execution time *pmf* (Fig. 3). Given the *pmf* of the backlog at time t, shrinking allows to compute the backlog at time $t' > t$ by shifting the *pmf* to the left by $t' - t$ units and by defining the probability of zero backlog be the sum of the probabilities defined for nonpositive values (Fig. 3). Here, t' represents a time instant *right before the next release time of a job that contributes to the P-level backlog*. Shrinking is equivalent to simulating the progression of time. Finally, the *pmf* of a job response time is initiated by computing the convolution of the execution time *pmf* of the job and the backlog at its release time. The additional contribution of preemptions occurring after the job is released is considered as follows. Suppose a higher priority job arrives at time t_k. The *pmf* of the job response time is split at t_k. The right-hand side is updated by convolution with the execution time *pmf* of the preempting job, and the two halves are then merged together. This process continues until the probability of a residual computation time for the job becomes zero before any other higher priority job is released. The stationary task response time *pmf* is obtained by averaging the response time *pmf*s of all the jobs within the hyperperiod.

V. Stochastic Analysis of Message Response Times

In this section, we present a stochastic analysis framework that approximates the response times for CAN messages. *None of the stochastic methods proposed so far*, including the one in [7] and [13], *covers the analysis of CAN periodic messages transmitted by nodes without clock synchronization.* The analysis provided in [7] and [13], for example, assumes knowledge of the (relative) activation times of tasks, and it estimates blocking delays due to the lack of preemption and release jitter in a pessimistic way. Other stochastic approaches assume Poisson arrivals and/or are not applicable to medium or light load conditions. Previous stochastic analysis of CAN systems in [15] and [16] focus on critical instant conditions, *without considering the probability of occurrence of a critical instant.* Also, [4] considers a model of random arrival streams, but only at one priority level. In [5], a stochastic analysis framework is proposed that provides probability distributions of message response times where the periods of messages are given by independent random variables. To our best knowledge, our method is *the first attempt to analyze message response time probabilities with nondeterministic message phasing*, which captures the nature of CAN-based systems where periodic messages are sent from ECUs with unsynchronized clocks. The following sections define the steps that are required for computing the *pmf* of the response time of an instance of message m_i on a generic ECU. The first step is to compute the interference from messages on remote ECUs.

A. A Modeling Abstraction for CAN Messages

We use one *characteristic interference message*, or simply *characteristic message*, for each remote node to model the interference caused by messages from it. Messages are queued by the TxTask. Hence, every message from the same node is queued with the same phase and its period is an integer multiple of the TxTask period. From the standpoint of message m_i, the queueings of higher priority instances from a remote node ECU_k $hp(P_i)_k = \{m_j | P_j < P_i \cap \Upsilon_j^{\mathrm{src}} = \mathrm{ECU}_k\}$ only happen

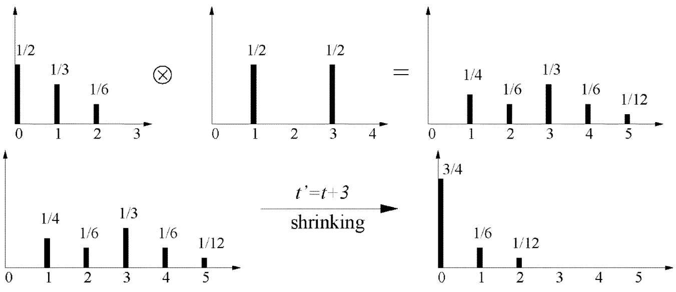


Fig. 3. Convolution and shrinking.

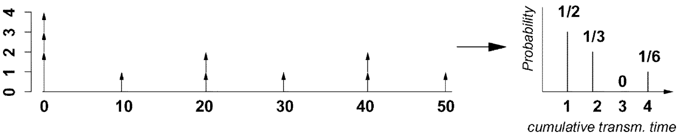


Fig. 4. An example of characteristic message transmission time.

at integer multiples of the greatest common divisor (gcd) of their periods. The characteristic message summarizes the effect of all such messages, with period equal to the gcd of their periods, nondeterministic transmission time, given that it represents sets of different length queued at different time instants and, with respect to remote message m_i, a random offset, because of the lack of synchronization among ECUs.

Fig. 4 provides an example of three remote messages with periods and transmission times equal to (60,2),(10,1),(20,1), respectively. During the hyperperiod [0,60), the message instances are queued at time instants $p \times 10, p = 0, 1, 2, 3, 4, 5$. At time 0, all messages are queued and the total transmission time is 4. The transmission times that are requested at the other time instants are 1, 2, 1, 2, 1, respectively. From the standpoint of a remote message m_i, the first subset that can produce interference is one of the six at random, given that the ECU clock offsets are uniformly distributed. Three time instants out of six contribute to a transmission time of 1 unit, two out of six, 2 units, and in one case out of six, 4 units. Thus, the *pmf* of the characteristic message transmission time is defined as $\mathbb{P}(1) = 1/2, \mathbb{P}(2) = 1/3, \mathbb{P}(4) = 1/6$.

Different from the example in the figure, CAN message transmission times $\mathcal{E}_j$ are nondeterministic because of the stuff bits. A simple model for computing the probability of a given number of stuff bits can be obtained by assuming equal probability of a zero or one value for each bit. Other models and methods can be found in [17].

Algorithm 1 computes the transmission time *pmf* $f_{\mathcal{E}_c}$ of the P_i-level characteristic message on a generic ECU_k. First, we identify the set of messages $hp(P_i)_k$, the lcm of their periods H_k^i, and the gcd of their periods T_c, the time interval that separates their queueing instants. Then, for each multiple of T_c within $[0, H_k^i)$, i.e., $t = pT_c, p = 0, 1, \ldots, H_k^i/T_c - 1$, the *pmf* of the total transmission time $\mathcal{E}[p]$ is calculated by adding (by convolution) the transmission time *pmf*s of the message instances queued at t. Finally, $f_{\mathcal{E}_c}$ is the average of the *pmf*s of $\mathcal{E}[p]$ at these time instants.

Algorithm 1: Calculate the transmission time pmf $f_{\mathcal{E}_c}$ of P_i-level characteristic message m_c for ECU_k

```
$hp(P_i)_k = \{m_j | P_j < P_i \cap \Upsilon_j^{\mathrm{src}} = \mathrm{ECU}_k\}$
$H_k^i = \mathrm{lcm}(\bigcup_{j \in hp(P_i)_k} T_j)$, $T_c = \mathrm{gcd}(\bigcup_{j \in hp(P_i)_k} T_j)$
for $p = 0, 1, \ldots, H_k^i/T_c - 1$ do
    $f_{\mathcal{E}[p]}(0) = 1.0$
    $t = p \times T_c$
    for each $j \in hp(P_i)_k$ do
        if $t \mathrm{mod} T_j = 0$ then
            $f_{\mathcal{E}[p]} = f_{\mathcal{E}[p]} \otimes f_{\mathcal{E}_j}$
        end if
    end for
end for
for $e = 0$ to $E_c^{\max}$ step $\tau$ do
    $f_{\mathcal{E}c}(e) = (T_c/H_k^i) \sum_{p=0}^{H_k^i/T_c - 1} f_{\mathcal{E}[p]}(e)$
end for
```

The characteristic message introduces inaccuracy in the analysis. If the response time of any instance of m_i is smaller than the minimum among the T_c of all remote nodes, then it will suffer interference from at most one set of message instances from any other node. Such set will be random, because of the random offsets among node clocks. Similarly, interference from the characteristic message will occur at most once and the amount of interference is the random transmission time of the characteristic message, which is the same as the interference from a random set of message instances for one of the pT_c time instants in the real case. However, when more than one interference is possible, in reality there is a correlation between the amount of interference provided by instances queued at consecutive cycles of the TxTask, which is lost in the characteristic message. In the example of Fig. 4, an interference of 4 units is *always* followed by an interference of one unit. With the characteristic message, there is a 1/36 chance that there will be another interference of 4 units. In general, even if the condition for absence of errors (response time of m_i always less than $\min\{T_c\}$) is true only for a (high priority) subset of all messages, our method retains sufficient accuracy for the evaluation of message response times even when this assumption is violated, as our experiments show.

The characteristic message allows a significant reduction in the size of the space to be analyzed. First, now only one message needs to be considered for each remote ECU. In addition, with respect to a remote message m_i, the only significant offset values of a characteristic message are in the range $[0, T_c)$, which is significantly smaller than $[0, H_k^i)$.

The second approximation/simplification in the model consists in changing the queueing model of remote characteristic messages from deterministic periodic with random initial offset [**(a)** in Fig. 5] into a deterministic offset, periodic activation times and random queueing jitter [**(b)** in Fig. 5].

The characteristic message m_c is queued with a random offset with respect to m_i, which depends on $\mathcal{O}_{\Upsilon_c^{\mathrm{src}}, \Upsilon_i^{\mathrm{src}}}$ (we

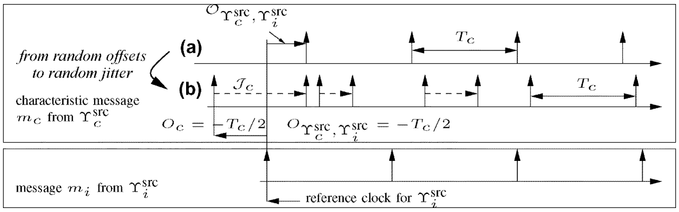

$\mathcal{O}_{\Upsilon_c^{\mathrm{src}},\Upsilon_i^{\mathrm{src}}}$
(a)
T_c
from random offsets to random jitter
(b)
$\mathcal{J}_c$
T_c
characteristic message m_c from $\Upsilon_c^{\mathrm{src}}$
$O_c = -T_c/2$
$\mathcal{O}_{\Upsilon_c^{\mathrm{src}},\Upsilon_i^{\mathrm{src}}} = -T_c/2$
message m_i from $\Upsilon_i^{\mathrm{src}}$
reference clock for $\Upsilon_i^{\mathrm{src}}$

Fig. 5. An example of characteristic message transmission time.

assume $O_c = -T_c/2$ and $O_i = 0$). In the approximate model, $\mathcal{O}_{\Upsilon_c^{\mathrm{src}},\Upsilon_i^{\mathrm{src}}} = 0$, and the randomness in the offsets among the ECU clocks is transformed into a constant zero offset among them, and random activation times for the characteristic messages, with offset $O_c = -T_c/2$ (to ensure a single interference on m_i if its response time is $< T_c/2$). At each activation, characteristic messages are delayed by a random jitter $\mathcal{J}_c$, uniformly distributed within the period T_c. For the jth instance $M_{c,j}$ of m_c, the queueing time is $\mathcal{Q}_{c,j} = O_c + (j-1)T_c + \mathcal{J}_c$. The initial offset $O_c = -T_c/2$ is also chosen to minimize the probability of multiple interferences from m_c to the first instance of m_i.

The introduction of random jitter in place of an initial random offset further introduces inaccuracy. In the model with random offsets, message transmissions are requested after exactly one period. This is not true for the characteristic message with a random release jitter (as shown in Fig. 5).

The formal definition of the P_i*-level characteristic message* m_c from $\Upsilon_c^{\mathrm{src}} = \Upsilon_k$ with respect to m_i is

- Period $T_c = \gcd(\bigcup_{j \in hp(P_i)_k} T_j)$, where $hp(P_i)_k = \{m_j | P_j < P_i \cap \Upsilon_j^{\mathrm{src}} = \Upsilon_k\}$.
- Initial offset $O_c = -T_c/2$, $\mathcal{O}_{\Upsilon_k,\Upsilon_i^{\mathrm{src}}} = 0$.
- Queueing jitter $\mathcal{J}_c$ uniformly distributed in $[0, T_c)$.
- Random transmission time $\mathcal{E}_c$, as shown in the example in Fig. 4 and Algorithm 1.
- Priority P_c higher than P_i.

In conclusion, to analyze the response time of a message m_i, we define an approximate system consisting of all the messages from the same node with priority higher than P_i. These messages are not abstracted using a characterization message, since they have known queueing instants with respect to m_i. In addition, we consider one characteristic message of level P_i for each of the other nodes. In the approximate system, every message m_j is modeled by $(T_j, O_j, \mathcal{J}_j, \mathcal{E}_j, P_j)$, where $\mathcal{J}_j$ is a random queueing jitter for characteristic messages and 0 for messages from the same node as m_i.

Theorem 1: If the bus utilization is less than 1, the approximate system model based on characteristic messages and the transformation of random offsets into deterministic offsets and random jitter is stable and admits a stationary distribution solution as in [13].

Proof: The proof is based on the results in [7] and [13] where the authors demonstrate stability provided that the model is periodic and the average resource utilization is less than 1.

Demonstrating the periodicity of the system model is simple. In our case, all messages are periodic, activated with known offsets and random jitter. The *lcm* of the message periods (or application hyperperiod) is the system period.

For the second part, the average utilization $\bar{U}_j$ of a message m_j is defined as the ratio between its expected transmission time $\bar{E}_j$ and its period T_j. First, we prove that the average utilization of a characteristic message m_c is equal to the sum of the average utilizations of the messages in the set $hp(P_i)_k$ it abstracts. Given that the execution time $\mathcal{E}_c$ of m_c is the average of the total transmission time $\mathcal{E}[p]$ at time instants $t = pT_c$, $p = 0, 1, \ldots, H_k^i/T_c - 1$ within one hyperperiod $[0, H_k^i)$, its expected value is

$$\bar{E}_c = \frac{T_c}{H_k^i} \times \sum_{p=0}^{H_k^i/T_c - 1} \bar{E}[p]. \qquad (1)$$

Each message $m_j \in hp(P_i)_k$, is queued H_k^i/T_j times in one hyperperiod $[0, H_k^i)$. Thus, $\sum_p \mathcal{E}[p] = \sum_{j \in hp(P_i)_k} (H_k^i/T_j)\mathcal{E}_j$, and the average value is

$$\sum_p \bar{E}[p] = \sum_{j \in hp(P_i)_k} \frac{H_k^i}{T_j} \bar{E}_j = H_k^i \times \sum_{j \in hp(P_i)_k} \bar{U}_j.$$

The average utilization $\bar{U}_c$ from m_c is

$$\bar{U}_c = \frac{\bar{E}_c}{T_c} = \frac{1}{T_c} \times \frac{T_c}{H_k^i} \times \sum_p \bar{E}[p] = \sum_{j \in hp(P_i)_k} \bar{U}_j. \qquad (2)$$

In our approximate system, when computing the latency *pmf* for m_i, we use one characteristic message of level P_i for each of the remote nodes. By (2), the utilization of each characteristic message is the sum of the utilization of the higher priority messages on its node. Hence, the average utilization of our approximate system is the same as in the original CAN system. The second part of the proof follows from the hypothesis that the original system has average utilization less than 1. □

However, the results obtained from our approximate system *are not always pessimistic* according to the definition in [13]. Our analysis may lead to optimistic probabilities of deadline misses, so design decisions based on the analysis result should be taken carefully, especially for time-critical systems where missing deadlines can lead to system failures. In general, our stochastic analysis should not be a tool for system validation, but rather for the comparison of architecture solutions and early estimates of performance. In most experimental cases, as in Figs. 7 and 8, the plots of the analysis results are below the simulation data, meaning that the analysis gives pessimistic results (especially for long latencies). However, bounding the error in either direction is not easy and we plan this work as a future extension.

The P*-level backlog* $\mathcal{W}_t^P$ *at time* t is the sum of the remaining transmission times of the queued message instances with priority higher than P. For each characteristic message instance $M_{p,q}$ we define the event "$M_{p,q}$ is queued after t" as $V_t^{p,q}$, or $V_t^{p,q} = (\mathcal{Q}_{p,q} > t)$, and its *complement* $\overline{V}_t^{p,q} = (\mathcal{Q}_{p,q} \leq t)$. Suppose t is a possible queueing time for n instances $M_{p_1,q_1}, M_{p_2,q_2}, \ldots, M_{p_n,q_n}$ with priorities higher than or equal to P. This set of instances is the P*-level message instance set at time* t, denoted as $I(P,t)$. For each instance $M_{p_k,q_k} \in I(P,t)$, the two mutually exclusive events $V_t^{p_k,q_k}$ and $\overline{V}_t^{p_k,q_k}$ divide the event space into two subspaces. When considering all instances in the set, there are totally 2^n possible combinations or event vectors, each representing a different combination of the random queueing times of the n instances with respect to t. These 2^n events represent the *P-level event space*

$$\begin{aligned}S(P,t) = \{&(V_t^{p_1,q_1}, V_t^{p_2,q_2}, \ldots, V_t^{p_n,q_n}),\\ &(V_t^{p_1,q_1}, V_t^{p_2,q_2}, \ldots, \overline{V}_t^{p_n,q_n}),\\ &\ldots, (\overline{V}_t^{p_1,q_1}, \overline{V}_t^{p_2,q_2}, \ldots, \overline{V}_t^{p_n,q_n})\}\end{aligned}$$

with total probability 1 defined for each priority level P and time t.

We define the *queueing pattern* at time t, denoted as $\mathcal{X}_t$, based on the possible queueing times of these message instances. $\mathcal{X}_t$ is a random variable defined over $S(P,t)$ with a known probability mass function. With a slight abuse of notation, $\mathcal{X}_t$ will also be used to indicate the *queueing pattern vector* at time t. For a queueing pattern $\mathcal{X}_t$, if the queueing event at t corresponding to the instance $M_{p,q}$ is $V_t^{p,q}$, then $\mathcal{X}_t$ is called a *positive queueing pattern* of $V_t^{p,q}$; otherwise, if $\overline{V}_t^{p,q}$ applies, it is a *negative queueing pattern* of $V_t^{p,q}$. The *complementary queueing pattern* $\overline{\mathcal{X}}_{t(p,q)}$ *with respect to* $M_{p,q}$ is defined by complementing the $V_t^{p,q}$ entry in $\mathcal{X}_t$.

We define a joint probability function of two random variables, $\mathcal{W}_t^P$, over the discretized $\mathbb{R}^+$, and $\mathcal{X}_t$ defined over $S(P,t)$, e.g.,

$$\begin{aligned}&f_{(\mathcal{W}_t^P,\mathcal{X}_t)}(w,(V_t^{p_1,q_1}, V_t^{p_2,q_2}, \ldots, V_t^{p_n,q_n}))\\ &\quad= \mathbb{P}(\mathcal{W}_t^P = w, \mathcal{X}_t = (V_t^{p_1,q_1}, V_t^{p_2,q_2}, \ldots, V_t^{p_n,q_n})).\end{aligned}$$

The original *pmf* for $\mathcal{W}_t^P$ can be reconstructed from the joint *pmf*s

$$f_{\mathcal{W}_t^P}(\cdot) = \sum_{x\in S(P,t)} f_{(\mathcal{W}_t^P,\mathcal{X}_t)}(\cdot, \mathcal{X}_t = x). \tag{3}$$

We are interested in computing the backlog $\mathcal{W}_t^{P_i}$ for an instance $M_{i,j}$ of m_i (actual message, not characteristic), where t is any instant after its queueing time ($t \geq \mathcal{Q}_{i,j}$) as a function of the interference set of $M_{i,j}$ or $I(P_i,t)$.

B. Stochastic Analysis of the Approximate System

In order to compute the *pmf* of the message response time in the approximate system, we first compute the stationary distribution of the backlog at the beginning of the hyperperiod; then, we compute the *pmf* of the backlog at the queueing time of each message instance, and finally, the *pmf* of the response time of each *message instance* within the hyperperiod. The message response time *pmf* is obtained by averaging the response time *pmf*s of all the instances in the hyperperiod.

1) Stationary Backlog Within the Hyperperiod: The example in Fig. 6 illustrates the method used to update the backlog at priority level P_i for m_i. The characteristic message instances, which represent the load from remote ECUs (such as $M_{p,q}$ in Fig. 6), are queued with random jitter in $[0,T_p)$. At any time t, we compute the backlog knowing its value at the previous tick $t-\tau$ and going through an intermediate step, an instant t^- arbitrarily close to t (this step is enlarged in the bubble on the top-left side of Fig. 6). Starting from the backlog at time $t-\tau$, we first update the backlog by shrinking (step **a** in Fig. 6) to time t^-, right before the possible queueing time of message instance $M_{p,q} \in I(P,t)$. By shrinking, we advance the time, accumulating in the origin (backlog equal to zero) all the probabilities of nonpositive backlogs. The probability that a message instance still to be activated is queued at t, $\mathbb{P}(\mathcal{Q}_{p,q} = t | \mathcal{Q}_{p,q} \geq t)$ can be easily computed as $1/n_t$, where n_t is the number of ticks from t to the latest possible queueing time $Q_{p,q}^{\max}$ for $M_{p,q}$.

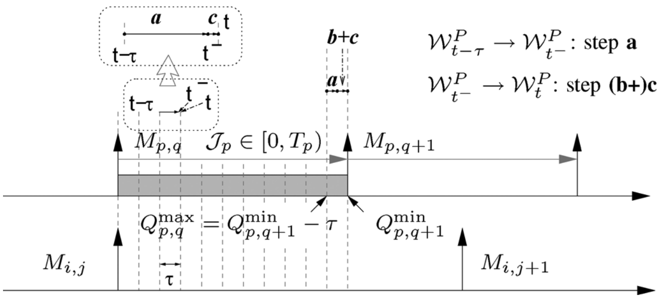

a
c
t
t−τ
t⁻
t−τ
t⁻
t
b+c
a
$\mathcal{W}_{t-\tau}^P \to \mathcal{W}_{t^-}^P$: step **a**
$\mathcal{W}_{t^-}^P \to \mathcal{W}_t^P$: step **(b+)c**
$M_{p,q}$
$\mathcal{J}_p \in [0,T_p)$
$M_{p,q+1}$
$Q_{p,q}^{\max} = Q_{p,q+1}^{\min} - \tau$
$Q_{p,q+1}^{\min}$
$M_{i,j}$
$M_{i,j+1}$
τ

Fig. 6. Updating the backlog in the discrete-time model.

Finally, we provide the rules to compute the backlog from t^- to t in two possible scenarios, as a result of the possible (probabilistic) queueing of characteristic messages from remote nodes at t. One possible scenario (step **c** in Fig. 6) is when the set of message instances that can contribute to the backlog remains the same as that of $t-\tau$. The other case is when at time $t = Q_{p,q+1}^{\min} = Q_{p,q}^{\max} + \tau$, there is at least one new instance $M_{p,q+1}$ that must be considered in place of $M_{p,q}$. In this case, an additional step, labeled as **b** in Fig. 6, must be performed after the shrinking and before considering the backlog update. The following subsections describe the procedure for updating the backlog in these two cases: the possible queueing of a message, and a change in the set of message instances.

Updating the Backlog for a Possible Queueing of a Message Instance: To find the probability of $\mathcal{W}_t^P = \mathcal{W}_{t^-}^P + e = w$ where e is either 0 or a possible contribution to the backlog by a message instance $M_{p,q}$, we consider the joint *pmf*s of $\mathcal{W}_t^P$ and the queueing patterns $\overline{V}_t^{p,q}$ and $V_t^{p,q}$, that is, $f_{(\mathcal{W}_t^P,\overline{V}_t^{p,q})}$ and $f_{(\mathcal{W}_t^P,V_t^{p,q})}$:

- Scenario (1): $\mathcal{W}_t^P = w$ and $\overline{V}_t^{p,q} : (\mathcal{Q}_{p,q} \leq t)$, that is, the backlog at time t at priority higher than P is w *and* the new message instance $M_{p,q}$ has been queued at time prior to or at t. Then, either $M_{p,q}$ contributed to the backlog at any time prior or equal to t^-, and in this case $e = 0$, that is, $M_{p,q}$'s transmission time $\mathcal{E}_p$ has been already included in $\mathcal{W}_{t^-}^P$, or $M_{p,q}$ is queued at t, and the sum of its transmission time $\mathcal{E}_p$ and the backlog at t^- must be equal to w.
- Scenario (2): $\mathcal{W}_t^P = w$ and $V_t^{p,q} : (\mathcal{Q}_{p,q} > t)$, that is, the backlog at time t at priority higher than P is w *and* the new message instance $M_{p,q}$ is queued after t and does not contribute to $\mathcal{W}_t^P$. Therefore, $\mathcal{E}_p$ is not part of the backlog at time t, and $\mathcal{W}_t^P = \mathcal{W}_{t^-}^P = w$.

The following theorem formulates the backlog update process.

Theorem 2: Consider a characteristic message instance $M_{p,q}$, possibly queued at time t, with priority higher than P. $\mathcal{W}_{t^-}^P$ is the P-level backlog immediately before t. The probability that the backlog at time t equals a given value w is computed with respect to the mutually exclusive events $\overline{V}_t^{p,q}$ and $V_t^{p,q}$

$$
\begin{aligned}
&\mathbb{P}(\mathcal{W}_t^P = w, \overline{V}_t^{p,q}) \\
&= \mathbb{P}(\mathcal{W}_{t^-}^P = w, \overline{V}_{t^-}^{p,q}) + \sum_{e=0}^{w} \mathbb{P}(\mathcal{W}_{t^-}^P = w - e, V_{t^-}^{p,q}) \\
&\quad \times \mathbb{P}(\mathcal{Q}_{p,q} = t | \mathcal{Q}_{p,q} \geq t) \times \mathbb{P}(\mathcal{E}_p = e) \\
&\mathbb{P}(\mathcal{W}_t^P = w, V_t^{p,q}) \\
&= \mathbb{P}(\mathcal{W}_{t^-}^P = w, V_{t^-}^{p,q}) \times \mathbb{P}(\mathcal{Q}_{p,q} > t | \mathcal{Q}_{p,q} \geq t) \\
&= \mathbb{P}(\mathcal{W}_{t^-}^P = w, V_{t^-}^{p,q}) \times (1 - \mathbb{P}(\mathcal{Q}_{p,q} = t | \mathcal{Q}_{p,q} \geq t)).
\end{aligned}
$$

The computation of the joint *pmf*s of the P-level backlog with respect to the two events $\overline{V}_t^{p,q}$ and $V_t^{p,q}$ can be performed as follows ($f_{\mathcal{V}_1} \otimes f_{\mathcal{V}_2}$ denotes the convolution of $f_{\mathcal{V}_1}$ and $f_{\mathcal{V}_2}$):

$$
\begin{aligned}
f_{(\mathcal{W}_t^P, \overline{V}_t^{p,q})} &= f_{(\mathcal{W}_{t^-}^P, \overline{V}_{t^-}^{p,q})} + \mathbb{P}(\mathcal{Q}_{p,q} = t | \mathcal{Q}_{p,q} \geq t) \\
&\quad \times (f_{(\mathcal{W}_{t^-}^P, V_{t^-}^{p,q})} \otimes f_{\mathcal{E}_p}) \qquad (4) \\
f_{(\mathcal{W}_t^P, V_t^{p,q})} &= (1 - \mathbb{P}(\mathcal{Q}_{p,q} = t | \mathcal{Q}_{p,q} \geq t)) \times f_{(\mathcal{W}_{t^-}^P, V_{t^-}^{p,q})}. \qquad (5)
\end{aligned}
$$

Proof: The proof can be found in [19]. In general, there may be more than one message instance queued at time t, therefore, we need to consider the joint *pmf*s of the backlog with all the queueing patterns $\mathcal{X}_t \in S(P, t)$. Using (4) and (5), we compute the joint *pmf* of the P-level backlog at t together with each pattern $\mathcal{X}_t$, after the possible queueing of a message instance $M_{p,q} \in I(P, t)$. In case $\mathcal{X}_t$ is a negative pattern of $V_t^{p,q}$, the *pmf* $f_{(\mathcal{W}_t^P, \mathcal{X}_t)}$ can be obtained by computing the convolution of the joint *pmf* defined with respect to the complementary queueing pattern $\overline{\mathcal{X}}_{t-(p,q)}$, $f_{(\mathcal{W}_{t^-}^P, \overline{\mathcal{X}}_{t-(p,q)})}$, with the transmission time *pmf* $f_{\mathcal{E}_p}$ of $M_{p,q}$, multiplied by $\mathbb{P}(\mathcal{Q}_{p,q} = t | \mathcal{Q}_{p,q} \geq t)$, and adding the result to the joint *pmf* $f_{(\mathcal{W}_{t^-}^P, \mathcal{X}_{t^-})}$ [by extension of (4)]. In case $\mathcal{X}_t$ is a positive pattern of $V_t^{p,q}$, $f_{(\mathcal{W}_t^P, \mathcal{X}_t)}$ is obtained by multiplying $f_{(\mathcal{W}_{t^-}^P, \mathcal{X}_{t^-})}$ with $1 - \mathbb{P}(\mathcal{Q}_{p,q} = t | \mathcal{Q}_{p,q} \geq t)$ [by extension of (5)]. This operation is called *cross-convolution* since the update of the joint *pmf* $f_{(\mathcal{W}_t^P, \mathcal{X}_t)}$ for a negative pattern $\mathcal{X}_t$ of $V_t^{p,q}$ is performed by adding a term resulting from the convolution of the transmission time *pmf* of $M_{p,q}$ with the joint*pmf* of the backlog and the complementary queueing pattern $\overline{\mathcal{X}}_{t-(p,q)}$ with respect to $M_{p,q}$, as described in Algorithm 2.

Algorithm 2: Cross-convolution: calculate P-level backlog after the possible queueing of $M_{p,q}$ at time t

```
for each negative pattern $\overline{\mathcal{X}}_t \in S(P, t)$ of $V_t^{p,q}$ do
    $\mathcal{X}_{t(p,q)} = \text{complementary pattern of } \overline{\mathcal{X}}_t$ wrt $M_{p,q}$
    $f_{(\mathcal{W}_t^P, \overline{\mathcal{X}}_t)} = f_{(\mathcal{W}_{t^-}^P, \overline{\mathcal{X}}_{t^-})} + \mathbb{P}(\mathcal{Q}_{p,q} = t | \mathcal{Q}_{p,q} \geq t) \times (f_{(\mathcal{W}_{t^-}^P, \mathcal{X}_{t-(p,q)})} \otimes f_{\mathcal{E}_p})$.
end for
for each positive pattern $\mathcal{X}_t \in S(P, t)$ of $V_t^{p,q}$ do
    $f_{(\mathcal{W}_t^P, \mathcal{X}_t)} = (1 - \mathbb{P}(\mathcal{Q}_{p,q} = t | \mathcal{Q}_{p,q} \geq t)) \times f_{(\mathcal{W}_{t^-}^P, \mathcal{X}_{t^-})}$
end for
```

Queueing Pattern Update: A special case occurs at the time instants when the sets of P-level message instances change. A characteristic message instance $M_{p,q}$ is replaced by $M_{p,q+1}$ at time $t = Q_{p,q+1}^{\min} = Q_{p,q}^{\max} + \tau$. The entries $V_{t^-}^{p,q}$ and $\overline{V}_{t^-}^{p,q}$, in the queueing pattern vector $\mathcal{X}_t$, must be replaced by two new queueing events $V_{t^-}^{p,q+1}$ and $\overline{V}_{t^-}^{p,q+1}$. The joint *pmf*s $f_{(\mathcal{W}_{t^-}^P, \mathcal{X}_{t^-})}$ for the new patterns can be defined starting from the consideration that $\mathbb{P}(\overline{V}_{t^-}^{p,q+1}) = \mathbb{P}(V_{t^-}^{p,q}) = 0$ and $\mathbb{P}(V_{t^-}^{p,q+1}) = \mathbb{P}(\overline{V}_{t^-}^{p,q}) = 1$. Hence, the *pmf* $f_{(\mathcal{W}_{t^-}^P, \mathcal{X}'_{t^-})}$ for each queueing pattern $\mathcal{X}'_{t^-}$ that includes $\overline{V}_{t^-}^{p,q+1}$, that is, $\mathcal{X}'_{t^-} = \{\ldots, \overline{V}_{t^-}^{p,q+1}, \ldots\}$ can be defined as $\mathbb{P}(\mathcal{W}_{t^-}^P = w, \mathcal{X}'_{t^-}) = 0, \forall w$. The values of $f_{(\mathcal{W}_{t^-}^P, \mathcal{X}''_{t^-})}$ for the other patterns that include the complement $V_{t^-}^{p,q+1}$ can be obtained by setting $f_{(\mathcal{W}_{t^-}^P, \mathcal{X}''_{t^-})} = f_{(\mathcal{W}_{t^-}^P, \mathcal{X}^*_{t^-})}$ where all the elements of $\mathcal{X}^*_{t^-}$ match the corresponding elements of $\mathcal{X}''_{t^-}$, except that $\overline{V}_{t^-}^{p,q}$ is replaced in $\mathcal{X}''_{t^-}$ by $V_{t^-}^{p,q+1}$.

Thus, given the stationary P-level backlog *pmf* at the beginning of the hyperperiod, the stationary P-level backlog *pmf* at any time in the hyperperiod can be computed by iteratively using the operations of shrinking, queueing pattern update (possibly) and cross-convolution.

2) Initial Blocking Time: Besides the delay from higher priority messages, a message is also subject to an initial blocking delay from lower priority messages due to the impossibility of preempting message transmissions. Given an instance $M_{i,j}$, we consider the set of all lower priority messages, denoted as $lp(P_i) = \{m_k | P_k > P_i\}$, which cause a blocking delay $\mathcal{B}_{i,j}$ to $M_{i,j}$ if they are transmitted at the time $M_{i,j}$ is queued. A blocking time of length $b > 0$ is caused if a message in $lp(P_i)$ has transmission time $\mathcal{E}_k > b$ and its transmission starts exactly $\mathcal{E}_k - b$ units before the queueing instant of $M_{i,j}$. If we assume that the transmissions of instances in $lp(P_i)$ are evenly distributed in the hyperperiod, then in our discrete-time system, the probability that one such instance transmission starts at exactly the right time tick in the hyperperiod is τ / T_k. The probability that $M_{i,j}$ waits for a time $b > 0$ is obtained by adding up all the contributions from the messages $m_k \in lp(P_i)$

$$
\mathbb{P}(\mathcal{B}_{i,j} = b) = \sum_{m_k \in lp(P_i)} \frac{\mathbb{P}(\mathcal{E}_k > b)}{\frac{T_k}{\tau}}. \qquad (6)
$$

Given these probabilities, the *pmf* $f_{\mathcal{B}_{i,j}}$ of $\mathcal{B}_{i,j}$ is easily computed and the blocking time is added (by convolution) to the backlog at the message queueing time. *Additional errors are introduced in this step, given that transmissions of lower priority messages are actually not uniformly distributed in the hyperperiod.* The initial blocking time is not included in the approximate system and does not affect its stability.

3) Message Response Time Calculation: The computation of the response time of $M_{i,j}$ requires, as a first step, the evaluation of its backlog at its queueing time. The other local messages are queued periodically at known instants, and the remote messages are represented by their P_i-level characteristic messages, queued periodically with random jitter. The backlog $\mathcal{W}^{P_i}$ is computed by applying the methods previously defined, and then the backlog at the queueing time of $M_{i,j}$ is updated to account for a possible initial blocking.

The earliest possible start time for $M_{i,j}$ is its queueing time $t = Q_{i,j}$. Starting from t and, considering the following times $t_k = t + k\tau$ ($k \in N^+$), we further decompose $f_{(\mathcal{W}_{t_k}^{P_i}, \mathcal{X}_{t_k})}$ into the joint *pmf*s $f_{(\mathcal{W}_{t_k}^{P_i}, \mathcal{S}_{i,j} < t_k, \mathcal{X}_{t_k})}$ and $f_{(\mathcal{W}_{t_k}^{P_i}, \mathcal{S}_{i,j} \geq t_k, \mathcal{X}_{t_k})}$, and the latter in $f_{(\mathcal{W}_{t_k}^{P_i}, \mathcal{S}_{i,j} = t_k, \mathcal{X}_{t_k})}$ and $f_{(\mathcal{W}_{t_k}^{P_i}, \mathcal{S}_{i,j} > t_k, \mathcal{X}_{t_k})}$.

At time t ($k = 0$), and for each queueing pattern $\mathcal{X}_t \in S(P_i, t)$, clearly $f_{(\mathcal{W}_t^{P_i}, \mathcal{S}_{i,j} \geq t, \mathcal{X}_t)} = f_{(\mathcal{W}_t^{P_i}, \mathcal{X}_t)}$ (the start time is no earlier than the queueing time). The probability that $M_{i,j}$ starts transmission immediately, i.e., $\mathbb{P}(\mathcal{S}_{i,j} = Q_{i,j} = t)$ equals the probability that the backlog of $M_{i,j}$ at t is zero

$$\mathbb{P}(\mathcal{S}_{i,j} = t) = \mathbb{P}(\mathcal{W}_t^{P_i} = 0) = \sum_{\mathcal{X}_t} \mathbb{P}(\mathcal{W}_t^{P_i} = 0, \mathcal{S}_{i,j} \geq t, \mathcal{X}_t).$$

Given that $\mathcal{W}_t^{P_i} = 0$ implies $\mathcal{S}_{i,j} = t$, and $\mathcal{W}_t^{P_i} = w$ with $w > 0$ implies $\mathcal{S}_{i,j} > t$, we have

$$\begin{aligned} \mathbb{P}(\mathcal{W}_t^{P_i} = 0, \mathcal{S}_{i,j} > t) &= 0 \\ \mathbb{P}(\mathcal{W}_t^{P_i} = w, \mathcal{S}_{i,j} > t) &= \mathbb{P}(\mathcal{W}_t^{P_i} = w, \mathcal{S}_{i,j} \geq t), \forall w > 0 \end{aligned}$$

which means that $f_{(\mathcal{W}_t^{P_i}, \mathcal{S}_{i,j} > t, \mathcal{X}_t)}(\mathcal{W}_t^{P_i} = 0) = 0$, and, for all $w > 0$, it is $f_{(\mathcal{W}_t^{P_i}, \mathcal{S}_{i,j} > t, \mathcal{X}_t)}(\mathcal{W}_t^{P_i} = w) = f_{(\mathcal{W}_t^{P_i}, \mathcal{S}_{i,j} \geq t, \mathcal{X}_t)}(\mathcal{W}_t^{P_i} = w)$ for each $\mathcal{X}_t \in S(M_{i,j}, t)$. Also, since $\mathcal{S}_{i,j} > t$ implies $\mathcal{W}_t^{P_i} > 0$, it is

$$\sum_{w} \mathbb{P}(\mathcal{W}_t^{P_i} = w, \mathcal{S}_{i,j} > t) = 1 - \mathbb{P}(\mathcal{S}_{i,j} = t). \tag{7}$$

The probability that $M_{i,j}$ starts transmission at the next time instants $t_k = t + k\tau$ can be computed by iteratively applying the following steps. Starting from $f_{(\mathcal{W}_t^{P_i}, \mathcal{S}_{i,j} > t, \mathcal{X}_t)}$, the compound *pmf*s of the backlog is updated to the next time point $t_1 = t + \tau$ by applying the methods defined in Section V-B1. In our discrete system, $\mathcal{S}_{i,j} > t$ is the same as $\mathcal{S}_{i,j} \geq t_1$, hence, $f_{(\mathcal{W}_{t_1}^{P_i}, \mathcal{S}_{i,j} \geq t_1)} = f_{(\mathcal{W}_{t_1}^{P_i}, \mathcal{S}_{i,j} > t)}$. Since the operations of cross-convolution, shrinking and queueing pattern update do not change the total probability of the backlog, thus

$$\sum_{w} \mathbb{P}(\mathcal{W}_{t_1}^{P_i} = w, \mathcal{S}_{i,j} \geq t_1) = 1 - \mathbb{P}(\mathcal{S}_{i,j} = t).$$

The probability that $M_{i,j}$ starts transmission at time t_1 equals the probability that its backlog is zero at t_1

$$\mathbb{P}(\mathcal{S}_{i,j} = t_1) = \mathbb{P}(\mathcal{W}_{t_1}^{P_i} = 0) = \sum_{\mathcal{X}_{t_1}} \mathbb{P}(\mathcal{W}_{t_1}^{P_i} = 0, \mathcal{S}_{i,j} \geq t_1, \mathcal{X}_{t_1}).$$

Similar to time instant t, we compute the compound *pmf* for the case $\mathcal{S}_{i,j} > t_1$ as $f_{(\mathcal{W}_{t_1}^{P_i}, \mathcal{S}_{i,j} > t_1, \mathcal{X}_{t_1})}(0) = 0$, and $\forall w > 0, f_{(\mathcal{W}_{t_1}^{P_i}, \mathcal{S}_{i,j} > t_1, \mathcal{X}_{t_1})}(w) = f_{(\mathcal{W}_{t_1}^{P_i}, \mathcal{S}_{i,j} \geq t_1, \mathcal{X}_{t_1})}(w)$. Iterating (7), the total probability of $f_{(\mathcal{W}_{t_1}^{P_i}, \mathcal{S}_{i,j} > t_1)}$ is

$$\sum_{w} \mathbb{P}(\mathcal{W}_{t_1}^{P_i} = w, \mathcal{S}_{i,j} > t_1) = 1 - \sum_{t_k \leq t_1} \mathbb{P}(\mathcal{S}_{i,j} = t_k).$$

We iteratively apply the above two steps for $t_k = t + k\tau$ until we get to time t_n when the compound backlog is null

$$\forall \mathcal{X}_n \in S(M_{i,j}, t_n), \forall w > 0, \mathbb{P}(\mathcal{W}_{t_n}^{P_i} = w, \mathcal{S}_{i,j} \geq t_n, \mathcal{X}_n) = 0.$$

This happens when, at time t_n, the total probability of the start time *pmf* accumulates to 1 (as outlined in Algorithm 3)

$$\sum_{k \leq n} \mathbb{P}(\mathcal{S}_{i,j} = t_k) = 1.$$

Algorithm 3: Calculate the start time pmf of $M_{i,j}$

```
$t = Q_{i,j}$
**for** each queueing pattern $\mathcal{X}_t \in S(M_{i,j}, t)$ **do**
    $f_{(\mathcal{W}_t^{P_i}, \mathcal{S}_{i,j} \geq t, \mathcal{X}_t)} = f_{(\mathcal{W}_{t-}^{P_i}, \mathcal{X}_t)}$
**end for**
$\forall k \geq Q_{i,j}, \mathbb{P}(\mathcal{S}_{i,j} = k) = 0$
**while** $\sum_{k=Q_{i,j}}^{t} \mathbb{P}(\mathcal{S}_{i,j} = k) < 1$ **do**
    **for** each queueing pattern $\mathcal{X}_t \in S(M_{i,j}, t)$ **do**
        $\mathbb{P}(\mathcal{S}_{i,j} = t) += \mathbb{P}(\mathcal{W}_t^{P_i} = 0, \mathcal{S}_{i,j} \geq t, \mathcal{X}_t)$
        $\forall w > 0, \mathbb{P}(\mathcal{W}_t^{P_i} = w, \mathcal{S}_{i,j} > t, \mathcal{X}_t) = \mathbb{P}(\mathcal{W}_t^{P_i} = w, \mathcal{S}_{i,j} \geq t, \mathcal{X}_t)$
    **end for**
    Update $f_{(\mathcal{W}_t^{P_i}, \mathcal{S}_{i,j} > t)}$ to get $f_{(\mathcal{W}_{t+\tau}^{P_i}, \mathcal{S}_{i,j} > t)}$, i.e., $f_{(\mathcal{W}_{t+\tau}^{P_i}, \mathcal{S}_{i,j} \geq t+\tau)}$
    $t = t + \tau$
**end while**
```

Once the *pmf* of the time $\mathcal{S}_{i,j}$ at which $M_{i,j}$ starts its transmission is known, the *pmf* of its finish time is obtained by simply adding its transmission time $\mathcal{E}_i$, i.e., performing a convolution on $f_{\mathcal{S}_{i,j}}$ and $f_{\mathcal{E}_i}$. The response time is finally computed by a further left shift of $Q_{i,j}$ time units

$$\begin{aligned} f_{\mathcal{F}_{i,j}} &= f_{\mathcal{S}_{i,j}} \otimes f_{\mathcal{E}_i} \\ \mathbb{P}(\mathcal{R}_{i,j} = t - Q_{i,j}) &= \mathbb{P}(\mathcal{F}_{i,j} = t), \forall t. \end{aligned} \tag{8}$$

The response time *pmf* $f_{\mathcal{R}_i}$ of m_i is the arithmetic average of the *pmf*s $f_{\mathcal{R}_{i,j}}$ of its instances $M_{i,j}$ inside the hyperperiod.

C. Experimental Results

In this section, we present results on one of the CAN buses of an experimental vehicle. The test set consists of 6 ECUs and 69 messages, as shown in Table I. T_i is in milliseconds, the message size is in bytes, and the transmission time distribution is calculated using the independent bit-stuffing model, as in [17]. All message offsets are zero. The bus rate is 500 kbps, and the maximum utilization of the system is 60.25%. We check the quality of our stochastic analysis by comparing its results with simulations. It takes less than 2 hours to analyze all messages, with a maximum of 7 min each with $\tau = 10$ μs. In our simulations, the relative phases among nodes are sampled with a granularity of 50 μs, leading to 3.2 $\times 10^{18}$ possible combinations. This space cannot be explored exhaustively, hence, we randomly

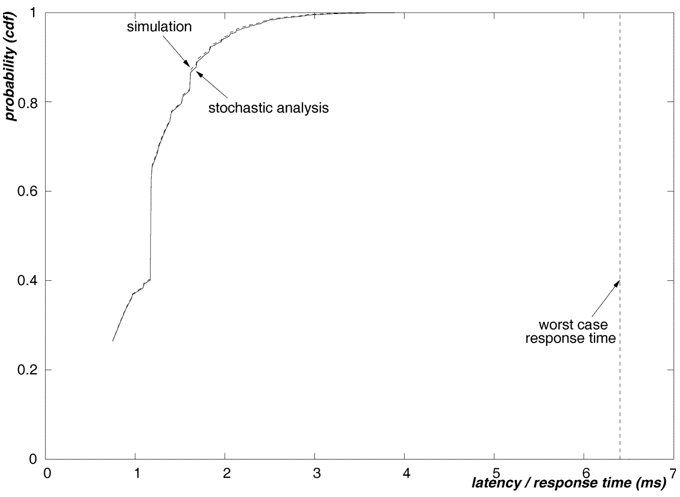


Fig. 7. The response time *cdf*s of a high-priority message m_{25} in Table I.

TABLE I
AN AUTOMOTIVE CAN SYSTEM WITH 6 ECUS AND 69 MESSAGES

m_i	Υ_i^{src}	T_i	size	P_i	m_i	Υ_i^{src}	T_i	size	P_i	m_i	Υ_i^{src}	T_i	size	P_i
m_1	ECU$_2$	10	8	1	m_{24}	ECU$_3$	25	6	24	m_{47}	ECU$_2$	50	4	47
m_2	ECU$_2$	10	8	2	m_{25}	ECU$_3$	25	7	25	m_{48}	ECU$_4$	100	8	48
m_3	ECU$_3$	5	4	3	m_{26}	ECU$_2$	20	8	26	m_{49}	ECU$_3$	100	8	49
m_4	ECU$_3$	10	7	4	m_{27}	ECU$_5$	25	8	27	m_{50}	ECU$_1$	100	8	50
m_5	ECU$_1$	10	4	5	m_{28}	ECU$_2$	20	8	28	m_{51}	ECU$_3$	100	1	51
m_6	ECU$_6$	10	8	6	m_{29}	ECU$_1$	25	3	29	m_{52}	ECU$_3$	100	8	52
m_7	ECU$_1$	100	8	7	m_{30}	ECU$_2$	10	5	30	m_{53}	ECU$_1$	100	4	53
m_8	ECU$_1$	100	2	8	m_{31}	ECU$_2$	20	8	31	m_{54}	ECU$_3$	100	1	54
m_9	ECU$_1$	100	3	9	m_{32}	ECU$_4$	10	8	32	m_{55}	ECU$_3$	100	1	55
m_{10}	ECU$_5$	25	8	10	m_{33}	ECU$_3$	10	8	33	m_{56}	ECU$_5$	100	8	56
m_{11}	ECU$_1$	100	2	11	m_{34}	ECU$_3$	10	8	34	m_{57}	ECU$_3$	100	7	57
m_{12}	ECU$_1$	20	4	12	m_{35}	ECU$_1$	25	7	35	m_{58}	ECU$_3$	100	1	58
m_{13}	ECU$_2$	100	4	13	m_{36}	ECU$_6$	25	6	36	m_{59}	ECU$_3$	100	1	59
m_{14}	ECU$_1$	100	4	14	m_{37}	ECU$_4$	50	8	37	m_{60}	ECU$_4$	100	3	60
m_{15}	ECU$_5$	100	8	15	m_{38}	ECU$_4$	50	8	38	m_{61}	ECU$_1$	100	8	61
m_{16}	ECU$_1$	10	6	16	m_{39}	ECU$_5$	50	8	39	m_{62}	ECU$_1$	100	1	62
m_{17}	ECU$_2$	100	7	17	m_{40}	ECU$_3$	50	5	40	m_{63}	ECU$_3$	100	4	63
m_{18}	ECU$_2$	100	8	18	m_{41}	ECU$_4$	50	8	41	m_{64}	ECU$_1$	100	8	64
m_{19}	ECU$_2$	50	7	19	m_{42}	ECU$_6$	50	8	42	m_{65}	ECU$_1$	100	1	65
m_{20}	ECU$_3$	10	8	20	m_{43}	ECU$_2$	50	8	43	m_{66}	ECU$_1$	100	1	66
m_{21}	ECU$_6$	10	8	21	m_{44}	ECU$_5$	100	8	44	m_{67}	ECU$_2$	50	8	67
m_{22}	ECU$_6$	25	8	22	m_{45}	ECU$_2$	25	2	45	m_{68}	ECU$_1$	100	1	68
m_{23}	ECU$_3$	25	6	23	m_{46}	ECU$_2$	50	4	46	m_{69}	ECU$_4$	100	8	69

chose 10^8 relative phases, for 10 hours of total running time. The stationary distribution of backlogs is obtained by an iterative approximation, that is, stopping when the *pmf*s of backlogs are close enough at the end of two consecutive hyperperiods. The stationary distribution of the message response times are calculated in the next hyperperiod. We show the results for two representative messages, one with relatively high-priority, and the other with low-priority, and compare the cumulative distribution functions (*cdf*s) of the stochastic analysis and simulation, along with the values from worst case analysis [6].

For message m_{25}, the worst case response time is slightly larger than the gcd of the periods (5 *ms*), but the analysis is quite accurate. The results start to deviate from the simulation when the worst case response time grows longer. Message m_{63} has a worst case response time significantly larger than the gcd of the periods. However, the distribution obtained from the analysis is still a good match to that of the simulation (Fig. 8). For low-priority messages, the probabilities computed by the stochastic analysis for long response times are larger than those computed by the simulation as the queueing times of the characteristic messages are a pessimistic approximation with respect to interferences (consecutive instances of a characteristic message can be separated by less than its period). Furthermore, as shown in both figures, the cumulative probability approaches 1 at response time values significantly smaller than the worst-case value. Finally, Fig. 9 shows the results of the message response time analysis of a complex message trace extracted from a product vehicle (not simulation data). The measured response time *cdf* of a low-priority message (CAN id > 0×500) is compared with the response time estimated by the stochastic analysis. The analysis is accurate despite the large worst-case message response time and the nonidealities of the actual implementation (possible priority inversions at the network adapter and significant copy times between the message queues and the TxObjects).

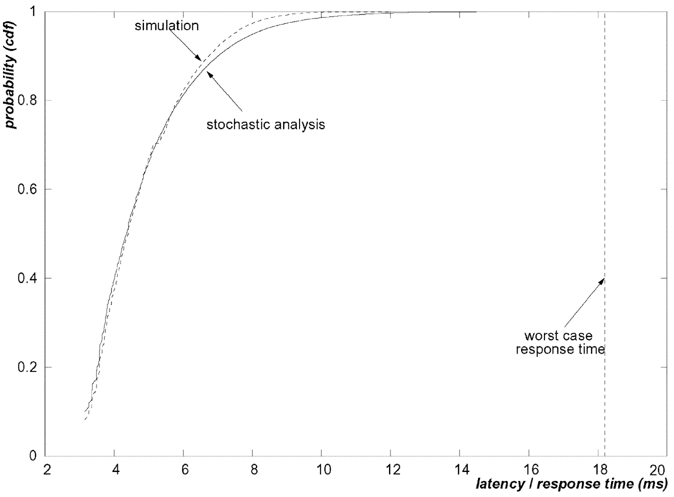


Fig. 8. The response time *cdf*s of a low-priority message m_{63} in Table I.

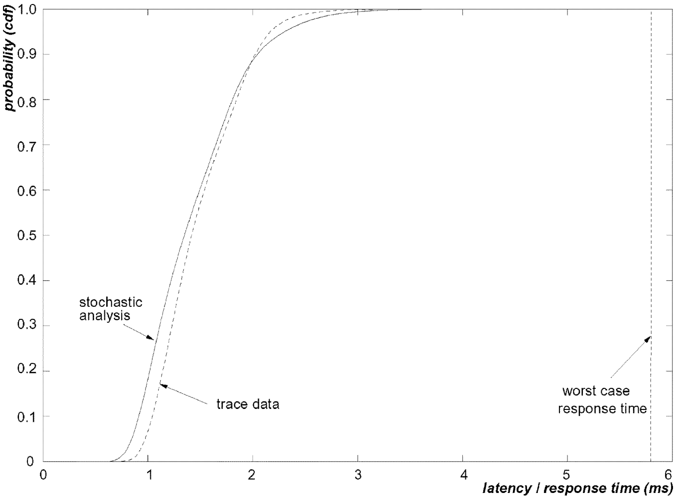


Fig. 9. The response time *cdf* of a low-priority message in a bus trace.

VI. STOCHASTIC ANALYSIS OF END-TO-END LATENCY

This section describes the stochastic analysis of the latency in the end-to-end propagation of information among periodically

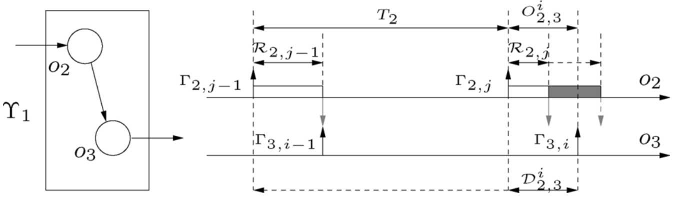


Fig. 10. Latency of task to task communication: local interaction.

activated tasks and messages. Given a path, its end-to-end latency analysis is divided into sections. We assume the response time *pmf* of each object in the path is known. With reference to the bottom part of Fig. 1, the end-to-end latency consists of an initial *sampling delay*, from the time the external event occurs (o_1 in Fig. 1) to the activation of the task (o_2) that reads it. Following, there are local and remote interactions, and finally, the latency is completed by the response time of the last task (o_9) in the chain.

A. Delays of Local Communication

In the case of local communication from task to task, or from task to message queueing, the latency from the activation of the sender to the activation of the receiver that reads its output depends on their relative phase and the response time of the sender. Consider the scenario of Fig. 10, which is part of the end-to-end path in Fig. 1. The local communication is between tasks o_2 and o_3, or τ_2 and τ_3. Suppose that there are n jobs of τ_3 in the hyperperiod, $\Gamma_{3,1}, \Gamma_{3,2}, \ldots, \Gamma_{3,n}$. Each time a job $\Gamma_{3,i}$ is activated and reads some data, we go back in the timeline, and find the job $\Gamma_{2,j}$ of τ_2 that produced the data, and then compute the delay between the activation events of $\Gamma_{2,j}$ and $\Gamma_{3,i}$. We use the notation $\mathcal{D}^i_{2,3}$ for the delay between the activation of $\Gamma_{3,i}$, and the activation of the job of τ_2 which produces the data consumed by $\Gamma_{3,i}$. $\mathcal{D}_{2,3}$ is the average delay associated with all the jobs of τ_3 in one hyperperiod. If the data produced by a job of τ_2 are overwritten by a new instance of τ_2 before they are consumed, the value is not propagated and the delay not considered in the distribution.

Since τ_2 and τ_3 are on the same node, the relative phase of each job $\Gamma_{3,i}$ with respect to the previous job $\Gamma_{2,j}$ of τ_2 is known and denoted as $O^i_{2,3}$. In case the response time of $\Gamma_{2,j}$ is less than or equal to $O^i_{2,3}$ (Fig. 10), the output of $\Gamma_{2,j}$ is consumed by the following job $\Gamma_{3,i}$ of τ_3 and the latency of this communication stage is $O^i_{2,3}$. Thus, the probability that the delay $\mathcal{D}^i_{2,3}$ is equal to $O^i_{2,3}$ is

$$\mathbb{P}(\mathcal{D}^i_{2,3} = O^i_{2,3}) = \mathbb{P}(\mathcal{R}_{2,j} \le O^i_{2,3}).$$

If the finish time of $\Gamma_{2,j}$ is larger than the activation time of $\Gamma_{3,i}$, the data consumed by $\Gamma_{3,i}$ is produced by $\Gamma_{2,j-1}$ provided that $\Gamma_{2,j-1}$ finishes execution before $\Gamma_{3,i}$ is activated

$$\mathbb{P}(\mathcal{D}^i_{2,3} = O^i_{2,3} + T_2) = \mathbb{P}(\mathcal{R}_{2,j} > O^i_{2,3}, \mathcal{R}_{2,j-1} \le O^i_{2,3} + T_2).$$

Also, we assume that the job response time is independent from other jobs of the same task, and from jobs of other tasks, thus the probability that $\mathcal{D}^i_{2,3}$ is equal to $O^i_{2,3} + T_2$ is

$$\begin{aligned}\mathbb{P}(\mathcal{D}^i_{2,3} &= O^i_{2,3} + T_2)\\ &= \mathbb{P}(\mathcal{R}_{2,j} > O^i_{2,3}) \times \mathbb{P}(\mathcal{R}_{2,j-1} \le O^i_{2,3} + T_2).\end{aligned}$$

In general, the data read by $\Gamma_{3,i}$ can be produced by any of the previous k instances from τ_2, until the probability for one such instance of τ_2 is zero. We need to try all the $k = 0, 1, \ldots, k_i$, where k_i is the first integer for which $R^{\max}_{2,j-k_i} \le O^i_{2,3} + k_i T_2$, where $R^{\max}_{2,j-k_i}$ is the worst case response time of job $\Gamma_{2,j-k_i}$. In other words, $\Gamma_{2,j-k_i}$ is the first job that is guaranteed to terminate before the activation of $\Gamma_{3,i}$ and to overwrite data produced by any previous job of τ_2

$$\begin{aligned}&\forall k = 0, 1, \ldots, k_i \text{ where } R^{\max}_{2,j-k_i} \le O^i_{2,3} + k_i T_2,\\ &\mathbb{P}(\mathcal{D}^i_{2,3} = O^i_{2,3} + kT_2) = \mathbb{P}(\mathcal{R}_{2,j-k} \le O^i_{2,3} + kT_2)\\ &\qquad \times \prod_{m=1}^{k} \mathbb{P}(\mathcal{R}_{2,j-k+m} > O^i_{2,3} + (k-m)T_2). \quad (9)\end{aligned}$$

The distribution of $\mathcal{D}_{2,3}$ is the average of the distributions of $\mathcal{D}^i_{2,3}$ for all the jobs $\Gamma_{3,i}$ of τ_3 within the hyperperiod, i.e.,

$$\forall d, \mathbb{P}(\mathcal{D}_{2,3} = d) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{P}(\mathcal{D}^i_{2,3} = d). \quad (10)$$

B. Delays of Remote Communication

In the case of remote communication, i.e., from a message to its receiving task, the relative phase of the task is independent from the message response time. We assume that for each instance of the message, its response time distribution is the same as the message response time. This is another approximation introduced for the purpose of simplifying the computations and a possible source of error.

Consider the communication between o_4 and o_5, i.e., message m_4 and task τ_5, which is part of the end-to-end path in Fig. 1. The relative phase $\mathcal{O}_{4,5}$ between τ_5 and the previously activated instance of m_4 is assumed to be uniformly distributed within $[0, T_4)$ and independent from the response time of m_4. $\mathcal{D}^i_{4,5}$ is the delay between the activation of a job $\Gamma_{5,i}$ of τ_5 and the activation of the instance of m_4 which produces the data consumed by $\Gamma_{5,i}$. Assume the relative phase $\mathcal{O}^i_{4,5}$ between $\Gamma_{5,i}$ and the previous message instance $M_{4,j}$ is a constant value d. Then, the probability distribution of $\mathcal{D}^i_{4,5}$ is [equation (9)]

$$\begin{aligned}&\forall k = 0, 1, \ldots, k_i \text{ where } R^{\max}_{4,j-k_i} \le d + k_i \times T_4,\\ &\mathbb{P}(\mathcal{D}^i_{4,5} = d + kT_4 | \mathcal{O}^i_{4,5} = d) = \mathbb{P}(\mathcal{R}_{4,j-k} \le d + kT_4 | \mathcal{O}^i_{4,5} = d)\\ &\qquad \times \prod_{m=1}^{k} \mathbb{P}(\mathcal{R}_{4,j-k+m} > d + (k-m)T_4 | \mathcal{O}^i_{4,5} = d). (11)\end{aligned}$$

Assuming the relative phase $\mathcal{O}^i_{4,5}$ is independent from the response time of any instance of m_4, thus $\mathbb{P}(\mathcal{R}_{4,j-k} \le d + kT_4 | \mathcal{O}^i_{4,5} = d) = \mathbb{P}(\mathcal{R}_{4,j-k} \le d + kT_4)$, and

$$\begin{aligned}\mathbb{P}(\mathcal{R}_{4,j-k+m} &> d + (k-m)T_4 | \mathcal{O}^i_{4,5} = d)\\ &= \mathbb{P}(\mathcal{R}_{4,j-k+m} > d + (k-m)T_4).\end{aligned}$$

Equation (11) can be simplified as

$$\begin{aligned}&\forall k = 0, 1, \ldots, k_i \text{ where } R^{\max}_{4,j-k_i} \le d + k_i \times T_4,\\ &\mathbb{P}(\mathcal{D}^i_{4,5} = d + kT_4 | \mathcal{O}^i_{4,5} = d) = \mathbb{P}(\mathcal{R}_{4,j-k} \le d + kT_4)\\ &\qquad \times \prod_{m=1}^{k} \mathbb{P}(\mathcal{R}_{4,j-k+m} > d + (k-m)T_4).\end{aligned}$$

Finally, we further simplify the analysis by assuming that for each instance $M_{4,j}$ of m_4, the *pmf* of its response time $\mathcal{R}_{4,j}$ is the same as the task response time $\mathcal{R}_4$ of m_4. $\mathcal{O}^i_{4,5}$ is uniformly distributed within $[0, T_4)$, thus the probability of it assuming any given value in the interval is τ/T_4. The probability function of $\mathcal{D}^i_{4,5}$ is

$$\begin{aligned}
&\forall d \in [0, T_4), \forall k = 0, 1, \ldots, k_i \\
&\quad \text{where } R^{\max}_{4,j-k_i} = R^{\max}_4 \le d + k_i \times T_4, \\
&\mathbb{P}(\mathcal{D}^i_{4,5} = d + kT_4) = \mathbb{P}(\mathcal{R}_4 \le d + kT_4) \\
&\qquad\qquad \times \prod_{m=1}^{k} \mathbb{P}(\mathcal{R}_4 > d + (k-m)T_4)) \frac{\tau}{T_4}.
\end{aligned}$$

$\mathcal{D}^i_{4,5}$ is independent from the instance index i, thus the right hand side remains the same for all jobs of τ_5 and it is the same as the distribution of $\mathcal{D}_{4,5}$, which is the average of $\mathcal{D}^i_{4,5}$ for all the jobs of τ_5 in the hyperperiod.

C. End-to-End Latency

The first contribution to the *end-to-end latency* $\mathcal{L}_{1,n}$ associated with the path $\Pi_{1,n}$ comes from the sampling delay $\mathcal{Z}_{1,2}$ from the activation of the source object o_1, which represents the external events, to the activation of the following object o_2. $\mathcal{Z}_{1,2}$ is assumed to be a uniform random variable within $[0, T_2)$. Following, $\mathcal{D}_{i-1,i}, \forall i = 2, \ldots, n$ is the delay from the activation of o_{i-1}, considered as the time instant when o_{i-1} consumes the data from its predecessor, to the following activation time of o_i after the completion of o_{i-1}, when its output is made available to the successor. The delay of data propagation from the time o_1 is activated to the time the final results are read for processing by o_n is the sum of $\mathcal{Z}_{1,2}$ and all the $\mathcal{D}_{i-1,i}$ for $i = 2 \ldots n$. At the end of the chain, the end-to-end latency is completed by an additional delay from the reading of the data by o_n at its activation time to its completion time, which is the response time $\mathcal{R}_n$ of o_n

$$\mathcal{L}_{1,n} = \mathcal{Z}_{1,2} + \sum_{i=2}^{n} \mathcal{D}_{i-1,i} + \mathcal{R}_n. \tag{12}$$

Assuming that $\mathcal{D}_{i-1,i}$, $\mathcal{Z}_{1,2}$, and $\mathcal{R}_n$ are independent from each other, $f_{\mathcal{L}_{1,n}}$ is the convolution of the *pmf*s of these terms

$$f_{\mathcal{L}_{1,n}} = f_{\mathcal{Z}_{1,2}} \bigotimes_{i=2}^{n} f_{\mathcal{D}_{i-1,i}} \otimes f_{\mathcal{R}_n}. \tag{13}$$

D. Experimental Results

We demonstrate the applicability of our approach with a case study derived from an experimental vehicle incorporating advanced active safety functions. The analysis focuses on one path with timing constraints. The architecture subsystem involved with the selected end-to-end path consists of 18 ECUs connected with two CAN buses at 500 kbps. A total of 71 tasks is executed on the ECUs, and 136 CAN messages are exchanged on the CAN buses. The worst-case execution time $E_i^{\max}$ is estimated for each task τ_i, and the task execution time is assumed to be uniformly distributed within $[(1/2)E_i^{\max}, E_i^{\max}]$. The selected path is composed of 6 tasks executing on 3 ECUs, and 2 CAN messages on the two buses. Fig. 11 shows the path in the example architecture platform. The tasks and messages involved are summarized in Table II, where the priorities are given as relative to the total number of objects in the brackets. The stochastic analysis of the case study was implemented in Java on a laptop with 1.6 GHz CPU and 1.5 G RAM and required 7.8 s of running time. Fig. 12 shows the results of the end-to-end latency *pmf*s of the example path. For our example, the worst-case latency value is 427.47 *ms* (with a probability less than 10^{-14}).

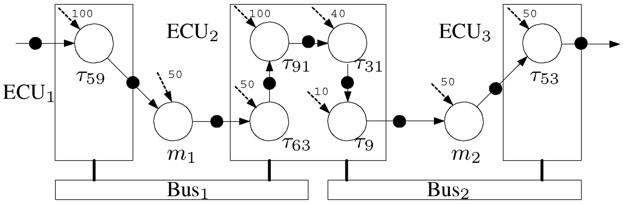


Fig. 11. A path of periodic tasks/messages in an example vehicle.

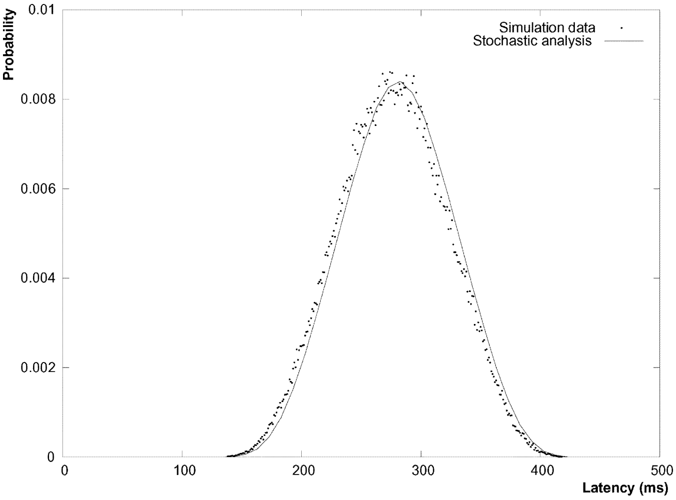


Fig. 12. The end-to-end latency *pmf*s of the path in the example vehicle.

TABLE II
CHARACTERISTICS OF TASKS AND MESSAGES IN THE END-TO-END PATH

o_i	$T_i(ms)$	O_i	P_i	$E_i^{\max}(ms)$	o_i	$T_i(ms)$	O_i	P_i	$E_i^{\max}$/size
τ_{59}	100	0	2(2)	2.331	τ_9	10	0	1(16)	2.331ms
τ_{63}	50	0	11(16)	13.314	τ_{53}	50	0	1(1)	0.252ms
τ_{91}	100	0	13(16)	5.656	m_1	50	0	13(106)	8bytes
τ_{31}	40	0	6(16)	2.163	m_2	50	0	16(30)	7bytes

We compared the results obtained with the stochastic analysis with simulation data obtained using a purposely developed system-level simulator. We simulated 4.5 s of system execution (three hyperperiods) for each phase configuration and tried approximately 50 millions phase configurations with a total running time of approximately 20 hours. The two sets of results (Fig. 12) are very close to each other, with a right shift (pessimism) of the stochastic analysis of approximately 5 *ms* (2% relative deviation). Overall, the most attractive feature of our stochastic analysis, when compared to a simulation-based approach, is its speed. With less than 8 s (as opposed to hours of simulations) it is possible to evaluate the latency of a path in an architecture configuration. This allows hundreds (or more) of different architecture options to be evaluated in an automated process.

VII. Conclusion

In this paper, we presented a stochastic analysis framework for the end-to-end latency of distributed real-time automotive systems. The analysis is based on previous work on periodic tasks with deterministic activation times. We proposed a novel stochastic analysis framework to compute CAN message response time *pmf*s in systems with unsynchronized ECUs and periodic messages. The method is based on the characterization of message interferences using a single message per node. We compose the response times of tasks and messages with the sampling delays caused by information passing in the communication-by-sampling model. The experimental sections compare the analysis results with simulation data for an experimental vehicle. The two sets of results are very close, but the stochastic analysis is significantly faster, enabling its use in an architecture exploration process.

Appendix
Alphabetic Notations

For a random variable, e.g., $\mathcal{E}_i$, we denote its average value as $\bar{E}_i$, its minimum and maximum value as $E_i^{\min}$ and $E_i^{\max}$.

- τ: granularity.
- τ_i: the ith task.
- $\Gamma_{i,j}$: the jth job of task τ_i.
- $\Pi_{i,j}$: path from object o_i to o_j.
- Υ_i: ECU or bus i.
- $\Upsilon_i^{\mathrm{src}}$: sending ECU of message m_i.
- $A_{i,j}$: arrival time of job $\Gamma_{i,j}$, or message instance $M_{i,j}$.
- $B_{i,j}$: blocking time of job $\Gamma_{i,j}$, or message instance $M_{i,j}$.
- $D_{i,j}$: delay between object o_i and o_j.
- $D_{i,j}^k$: delay of the data consumed by object instance $o_{j,k}$ and the instance of o_i who produces this data.
- E: set of edges representing the *data signals* among tasks and messages.
- E_i: execution time of τ_i, or transmission time of m_i.
- $f_{\mathcal{W}_t^P}$: *pdf* or *pmf* of random variable $\mathcal{W}_t^P$.
- $f_{(\mathcal{W}_t^P,\mathcal{X}_t)}$: joint *pdf* or *pmf* of random variable $\mathcal{W}_t^P$ and $\mathcal{X}_t$.
- $F_{i,j}$: finish time of job $\Gamma_{i,j}$, or message instance $M_{i,j}$.
- G_k^P: P-level backlog at the beginning of the k-th hyperperiod, i.e., $W_{(k-1)H}^P$.
- H: system hyperperiod.
- H_k^i: lcm of the periods of the objects in $hp(P_i)_{\Upsilon_k}$.
- $hp(P)$: $\{\tau_i(m_i) | P_i < P\}$, the set of tasks (messages) with priority higher than P.
- $hp(P)_k$: $\{\tau_i(m_i) | P_i < P \cap \Upsilon_i = \Upsilon_k\}$, the set of tasks (messages) with priority higher than P from ECU Υ_k.
- $I(P,t)$: P-level jobs or message instances at time t.
- J_i: release jitter of τ_i, or queueing jitter of m_i.
- $L_{i,j}$: end-to-end latency associated with path $\Pi_{i,j}$.
- $lp(P)$: $\{\tau_i(m_i) | P_i > P\}$, the set of lower priority tasks (messages).
- m_i: the ith message.
- $M_{i,j}$: the jth message instance of m_i.
- o_i: the ith object, i.e., task τ_i or message m_i.
- O_i: initial phase of task τ_i, or message m_i.
- $O_{i,j}^k$: relative phase of $o_{j,k}$ and the previous instance of o_i.
- $O_{\Upsilon_i,\Upsilon_j}$: clock difference between ECUs Υ_i and Υ_j.
- $O_{\Upsilon_i,j}$: relative phase of m_j with respect to Υ_i.
- P_i: priority of task τ_i, or message m_i.
- $Q_{i,j}$: release time of $\Gamma_{i,j}$, or queueing time of $M_{i,j}$
- $R_{i,j}$: response time of job $\Gamma_{i,j}$, or message instance $M_{i,j}$.
- $S(P,t)$: P-level event space at time t.
- T_i: period of task τ_i, or message m_i.
- U: utilization.
- U_i: utilization of object o_i.
- V: set of vertices representing the *tasks* and *messages*.
- $V_t^{i,j}$: event $\mathcal{Q}_{i,j} > t$.
- $\overline{V}_t^{i,j}$: the complement event of $V_t^{i,j}$, i.e., event $\mathcal{Q}_{i,j} \leq t$.
- W_t^P: P-level backlog at time t.
- $W_t^{M_{i,j}}$: backlog of message instance $M_{i,j}$ at time t.
- X_t: queueing pattern at time t.
- $\overline{X}_{t(i,j)}$: the complement queueing pattern of X_t with respect to $\Gamma_{i,j}$ or $M_{i,j}$.

References

[1] *Autosar Consortium Web Page*, [Online]. Available: http://www.autosar.org

[2] *ISO 11898–1. Road Vehicles – Interchange of Digital Information – Controller Area Network (can) for High-Speed Communication*, ISO Standard-11898, Nov. 1993, International Standards Organization (ISO).

[3] *Osek/vdx Operating System Specification Version 2.2.3,* Feb. 2005. [Online]. Available: http://www.osek-vdx.org

[4] I. Broster and A. A. Burns, "Random arrivals in fixed priority analysis," in *Proc. 1st Int. PARTES Workshop*, 2004.

[5] L. Cucu, "Preliminary results for introducing dependent random variables in stochastic feasibility analysis on CAN," in *Proc. 7th IEEE Int. WFCS Workshop, WIP Session*, May 2004, pp. 271–274.

[6] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien, "Controller area network (can) schedulability analysis: Refuted, revisited and revised," *Real-Time Systems*, vol. 35, no. 3, pp. 239–272, 2007.

[7] J. L. Díaz, D. F. García, K. Kim, C. G. Lee, L. Lo Bello, J. M. López, S. L. Min, and O. Mirabella, "Stochastic analysis of periodic realtime systems," in *Proc. 23rd IEEE Real-Time Syst. Symp.*, 2002, pp. 289–302.

[8] M. K. Gardner, "Probabilistic analysis and scheduling of critical soft real-time systems," Ph.D. dissertation, Dept. Comput. Sci., Univ. Illinois at Urbana-Champaign, Urbana, IL, 1999.

[9] J. Kim and K. G. Shin, "Execution time analysis of communicating tasks in distributed systems," *IEEE Trans. Comput.*, vol. 45, pp. 572–579, May 1996.

[10] J. P. Lehoczky, "Real-time queueing theory," in *Proc. 17th IEEE Real-Time Syst. Symp.*, Dec. 1996, pp. 186–195.

[11] J. P. Lehoczky, L. Sha, and Y. Ding, "The rate-monotonic scheduling algorithm: Exact characterization and average case behavior," in *Proc. 10th IEEE Real-Time Syst. Symp.*, Dec. 1989, pp. 166–171.

[12] W. Lei, Z. Wu, and M. Zhao, "Worst-case response time analysis for osek/vdx compliant real-time distributed control systems," in *Proc. 28th Annu. Int. Comput. Softw. Appl. Conf. (COMPSAC'04)*, 2004, pp. 148–153.

[13] J. M. López, J. L. Díaz, J. Entrialgo, and D. García, "Stochastic analysis of real-time systems under preemptive priority-driven scheduling," *Real-Time Syst.*, vol. 40, no. 2, pp. 180–207, 2008.

[14] S. Manolache, P. Eles, and Z. Peng, "Schedulability analysis of multiprocessor real-time applications with stochastic task execution times," in *Proc. 2002 IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD '02)*, 2002, pp. 699–706.

[15] N. Navet, Y.-Q. Song, and F. Simonot, "Worst-case deadline failure probability in real-time applications distributed over controller area network," *J. Syst. Architecture*, vol. 46, no. 7, pp. 607–617, 2000.

[16] T. Nolte, H. Hansson, and C. Norström, "Probabilistic worst-case response-time analysis for the controller area network," in *Proc. 9th IEEE Real-Time and Embedded Technol. Appl. Symp. (RTAS)*, May 2003, pp. 200–207.

[17] T. Nolte, H. Hansson, C. Norström, and S. Punnekkat, "Using bit-stuffing distributions in CAN analysis," in *Proc. IEEE/IEE Real-Time Embedded Syst. Workshop*, Dec. 2001.

[18] J. C. Palencia and M. Gonzáles Harbour, "Schedulability analysis for tasks with static and dynamic offsets," in *Proc. 19th IEEE Real-Time Syst. Symp. (RTSS'98)*, Dec. 1998, pp. 26–37.

[19] H. Zeng, "Probabilistic timing analysis of distributed real-time automotive systems," Ph.D. dissertation, EECS Dept., Univ. California, Berkeley, CA, 2008.

[20] ISO/DIS 26262–1 Road Vehicles – Functional Safety ISO. [Online]. Available: www.iso.org

Haibo Zeng received the B.E. and M.E. degrees in electrical engineering from Tsinghua University, Beijing, China, and the Ph.D. degree in electrical engineering and computer sciences from University of California at Berkeley, Berkeley.

He is a researcher at General Motors R&D. His research interests cover design methodology, analysis, and optimization for real-time systems and embedded systems.

Marco Di Natale (M'03) received the Ph.D. degree from Scuola Superiore Sant'Anna, Pisa, Italy, in 1991

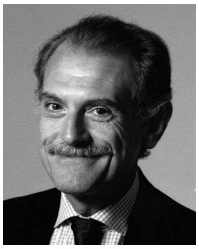
He is an Associate Professor at the Scuola Superiore Sant'Anna in which he held a position as Director of the Real-Time Systems (ReTiS) Laboratory from 2003 to 2006. He has been a Visiting Researcher at the University of California, Berkeley, in 2006 and 2008. He was selected in 2006 by the Italian Ministry of Research as the national representative in the mirror group of the ARTEMIS European Union Technology platform. He has been a researcher in the area of real-time systems and embedded systems for more than 15 years, being author or coauthor of more than 80 scientific papers.

Prof. Di Natale was a winner of three Best Paper Awards and the Archie T. Colwell Award. He has served as Program Committee member and has been organizer of tutorials and special sessions for the main conferences in the area, including the Real-time Systems Symposium, the IEEE/ACM Design Automation Conference (DAC), the Design Automation and Test in Europe (DATE), and the Real-Time Application Symposium. He also served as Track Chair for the RTAS Conference and the Automotive Track of the 2010 DATE Conference. He has been Associate Editor for the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN and is currently on the Editorial Board of the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS.

Paolo Giusto has 21 years of industrial experience in the area of methods and tools for the design, analysis, and optimization of real time systems.

Currently, he is a Staff Senior Researcher at General Motors, Advanced Technology, Silicon Valley Office, Palo Alto, CA, leading a project aimed at introducing novel design methodologies, methods, and tools into the GM electronics and software architecture development cycle. The methodologies, methods, and tools are aimed at shortening the time to market from the conceptual idea to the implementation in the real vehicle by enabling designers to quickly analyze design alternatives earlier in the development cycle. In addition, he is the responsible liaison between GM R&D and the research activities with University of California, Berkeley. He has worked in several areas ranging from research to product development to technical marketing and has authored or coauthored more than 35 scholarly papers and a book.

Mr. Giusto is serving as the Chairman and Organizer of the Society of Automotive Engineers Session 318 (System Level Design Tools and Methods) at the 2010 SAE World Congress.



Alberto Sangiovanni-Vincentelli (F'82) received the electrical engineering and computer science degree "Dottore in Ingegneria" *summa cum laude* from the Politecnico di Milano, Milano, Italy, in 1971.

He holds the Edgar L. and Harold H. Buttner Chair of Electrical Engineering and Computer Sciences at the University of California at Berkeley. He has been on the Faculty since 1976. From 1980 to 1981, he spent a year as a Visiting Scientist at the Department of Mathematical Sciences, IBM T. J. Watson Research Center. In 1987, he was Visiting Professor at the Massachusetts Institute of Technology (MIT). He has held a number of visiting professor positions at Italian Universities, including Politecnico di Torino, Universita' di Roma, La Sapienza, Universita' di Roma, Tor Vergata, Universita' di Pavia, Universita' di Pisa, and Scuola di Sant'Anna. He was a co-founder of Cadence and Synopsys, the two leading companies in the area of Electronic Design Automation. He is the Chief Technology Adviser of Cadence. He is a member of the Board of Directors of Cadence and the Chair of its Technology Committee, UPEK, a company he helped spinning off from ST Microelectronics, Sonics, and Accent, an ST Microelectronics-Cadence joint venture he helped founding. He was a member of the HP Strategic Technology Advisory Board, and is a member of the Science and Technology Advisory Board of General Motors and of the Scientific Council of the Tronchetti Provera foundation and of the Snaidero Foundation. He consulted for many companies including Bell Labs, IBM, Intel, United Technologies Corporation, COMAU, Magneti Marelli, Pirelli, BMW, Daimler-Chrysler, Fujitsu, Kawasaki Steel, Sony, ST, United Technologies Corporation, and Hitachi. He was an advisor to the Singapore Government for microelectronics and new ventures. He consulted for Greylock Ventures and for Vertex Investment Venture Capital funds. He is a member of the Advisory Board of Walden International, Sofinnova and Innogest Venture Capital funds and a member of the Investment Committee of a novel VC fund, Atlante Ventures, by Banca Intesa/San Paolo. He is the founder and Scientific Director of the Project on Advanced Research on Architectures and Design of Electronic Systems (PARADES), a European Group of Economic Interest supported by Cadence, Magneti-Marelli and ST Microelectronics. He is a member of the Advisory Board of the Lester Center for Innovation of the Haas School of Business and of the Center for Western European Studies and is a member of the Berkeley Roundtable of the International Economy (BRIE). He is a member of the High-Level Group, of the Steering Committee, of the Governing Board and of the Public Authorities Board of the EU Artemis Joint Technology Initiative. He is member of the Scientific Council of the Italian National Science Foundation (CNR) and Member of the Board of Directors of Rete Ventures. He is an author of over 800 papers, 15 books, and 3 patents in the area of design tools and methodologies, large-scale systems, embedded systems, hybrid systems and innovation.

Dr. Sangiovanni-Vincentelli is a Member of the National Academy of Engineering, the highest honor bestowed upon a U.S. engineer, since 1998. He received the Distinguished Teaching Award of the University of California in 1981. He received the Worldwide 1995 Graduate Teaching Award of the IEEE (a Technical Field Award for "Inspirational Teaching of Graduate Students"). In 2002, he was the recipient of the Aristotle Award of the Semiconductor Research Corporation. He has received numerous research awards including the Guillemin-Cauer Award (1982–1983), the Darlington Award (1987–1988) of the IEEE for the best paper bridging theory and applications, and two awards for the best paper published in the IEEE Transactions on CAS and CAD, five best paper awards and one best presentation awards at the Design Automation Conference, other best paper awards at the Real-Time Systems Symposium and the VLSI Conference. In 2001, he was given the Kaufman Award of the Electronic Design Automation Council for "Pioneering Contributions to EDA." In 2008, he was awarded the IEEE/RSE Wolfson James Clerk Maxwell Medal "for groundbreaking contributions that have had an exceptional impact on the development of electronics and electrical engineering or related fields" with the following citation: "for pioneering innovation and leadership in electronic design automation that have enabled the design of modern electronics systems and their industrial implementation." In 2009, he received the first ACM/IEEE A. Richard Newton Technical Impact Award in Electronic Design Automation to honor persons for an outstanding technical contribution within the scope of electronic design automation.