Comparison of the Temporal Performance of Physical and Virtual CAN Networks

R. Obermaisser, P. Peti
Vienna University of Technology, Austria

Abstract— **The automotive industry is a the verve to deploy computer systems not only for safety-related and comfort functionality, but for safety-critical by-wire systems. Time-triggered networks not only provide the communication infrastructure for these safety-critical application subsystems, but also permit event-triggered CAN communication via virtual CAN networks. Thus, there is the possibility to eliminate physical CAN networks, which leads to cost reductions and reliability improvements.**

However, a prerequisite for the replacement of physical CAN networks through virtual CAN networks is the ability to provide the temporal performance (e.g., communication latencies, bandwidth) required by existing CAN-based application software. This paper provides experimental results that demonstrate that virtual CAN networks can not only support the temporal performance of legacy applications, but go beyond the limitations of its physical counterpart by offering bandwidths above 1 Mbps and less latency jitter.

We perform measurements of the temporal performance of physical and virtual CAN networks with a framework that comprises an implementation of virtual CAN networks in the Time-Triggered Architecture and a Matlab/Simulink-based simulation of a physical CAN network. In order to compare the temporal performance, we use message sets provided by the automotive industry as inputs to both the simulation of physical CAN and the implementation of a virtual CAN network.

I. INTRODUCTION

Virtual CAN networks in the Time-Triggered Architecture (TTA) are a solution for layering event-triggered CAN communication on top of underlying time-triggered communication services [1]. Virtual CAN networks are of high industrial relevance for the automotive domain, since the CAN protocol is widely used in present day automotive networks for powertrain and body/comfort functions. Despite the use of time-triggered architectures in future by-wire cars, CAN is likely to remain as a communication protocol for non-safety critical functions due to the higher flexibility and average performance. Even for safety-related functions, CAN-based legacy applications will not be replaced instantly. In this context, the virtual CAN networks in a time-triggered environment enable the integration of applications for which a CAN communication service is preferable, as well as the integration of CAN legacy applications into a time-triggered computing platform.

In this paper, we demonstrate the ability to provide an authentic CAN communication service, as required for the integration of CAN-based legacy applications. This paper compares the temporal performance of physical and virtual CAN networks based on message sets from real-world automotive systems. We present experimental results of an implementation of virtual CAN networks in the Time-Triggered Architecture. In addition, we use the message sets as an input to a Matlab/Simulink-based simulation of a physical CAN network. The measured message transmission latencies and handled bandwidths show that virtual CAN networks are capable of supporting legacy integration. Furthermore, virtual CAN networks offer to newly-developed CAN-based applications a communication infrastructure that exceeds limitations of a physical CAN network (e.g., more bandwidth, smaller latencies, less jitter).

This paper is structured as follows. Section II discusses the temporal properties of a physical CAN network and defines the required temporal performance of a virtual CAN network in order to support the reuse of CAN-based legacy application. The construction of virtual networks as part of an integrated architecture for event-triggered and time-triggered control is the focus of Section III. In Section IV, we describe the measurement framework comprising an implementation of a virtual CAN network and a Matlab/Simulink-based simulation of a physical CAN network. We present the experimental results for real-word automotive message sets, as well as synthetic message sets in Section V. The interpretation of the collected data occurs in Section VI. The paper finishes with a conclusion in Section VII.

II. TEMPORAL PROPERTIES OF A PHYSICAL CAN NETWORK

CAN belongs to the class of event-triggered communication protocols. Messages exchanges are initiated after the occurrence of significant events, such as a transmission requests from the application. It uses a broadcast bus with Carrier Sense Multiple Access Collision Avoidance (CSMA/CA) for medium access control [2]. The bit transmission takes two possible representations, namely a recessive state (logical 0) and a dominant state (logical 1) that occurs when at least one sender transmits a dominant bit. Bus access conflicts are resolved by observing the message identifier bits on the bus-line. If a sender observes a dominant bit while transmitting a recessive bit, the sender gives up from transmitting and starts to receive incoming data. The sender transmitting the message with the lowest identifier will succeed and acquire bus access.

Consequently, the transmission latency of a message in a CAN network depends on the bus state at the point in time of the transmission request and the set of higher priority messages competing for bus access. The smallest guaranteed message transmission latency can be established for the highest priority message. In the worst-case, the transmission of the highest priority message is delayed by another CAN message of maximum size, which has progressed beyond the arbitration field when the sending of the message is requested. As derived in [3], the upper bound for the transmission latency R_h of the highest priority message h is:

$$R_h = J_h + \max_{\forall k \in lp(h)} (C_k) + C_h \tag{1}$$

where J_h is the queuing jitter of the message h, which depends on task response times. The second summand denotes the longest time message h can be delayed by lower priority messages $lp(h)$. C_h is the longest time required to physically send the messages at the CAN network. C_h depends on the bit time of the bus, the message size (0–8 bytes), and the message format (standard frame or extended frame).

In the presence of communication faults, inaccessibility times [4] must be considered for the determination of message transmission latencies. For example, handling of node failures is performed with error counters by recording receive and transmit errors. A threshold is defined for entering the error passive mode. In this mode, a node must wait for a minimum idle time on the bus before starting a transmission. If bus contention is low, this strategy results in the interleaving of correct and invalid messages. If a node's error counter exceeds a second threshold, the node enters the bus-off state. Under the assumption that faulty nodes reach the bus-off state, the worst-case inaccessibility time at 1 Mbps is bounded by 2.5 ms [5].

III. Integrated Architecture for Time-Triggered and CAN-based Applications

By providing CAN communication services in the Time-Triggered Architecture (TTA), we establish an integrated architecture for event-triggered and time-triggered control paradigms [6]. This integrated architecture is designed for mixed-criticality systems up to the highest criticality class (e.g., level A in RTCA DO-178B [7] or SIL4 in EN ISO/IEC 61508 [8]). The integrated architecture supports two subsystems, namely a time-triggered subsystem and a CAN subsystem. Safety-critical functionality is always realized in the time-triggered subsystem, while the CAN subsystem offers to non safety-critical applications an execution environment with high flexibility and average performance. The underlying design decision of using the time-triggered control paradigm for safety-critical functionality conforms to widely accepted requirements for the infrastructure of safety-critical real-time systems [9], [10].

As depicted in Figure 1 the integrated architecture is a waist-line architecture, which distinguishes between a minimal set of *core services* and *high-level services* that build on top of the core services. The core services are a common basis for the time-triggered subsystem and the high-level services of the CAN subsystem. The core services include a predictable time-triggered message transport service, fault tolerant clock synchronization,

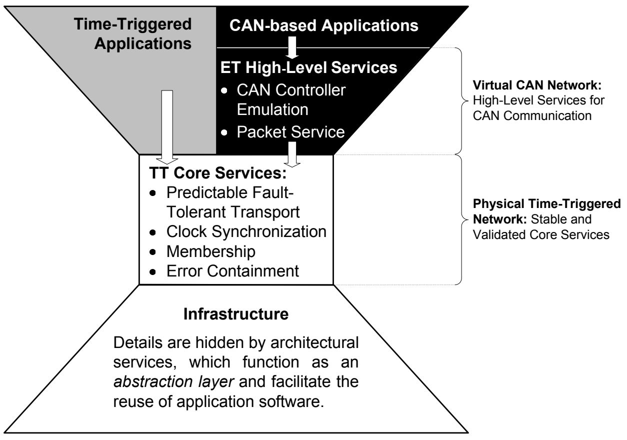

Time-Triggered Applications
CAN-based Applications
ET High-Level Services
• CAN Controller Emulation
• Packet Service
TT Core Services:
• Predictable Fault-Tolerant Transport
• Clock Synchronization
• Membership
• Error Containment
Infrastructure
Details are hidden by architectural services, which function as an *abstraction layer* and facilitate the reuse of application software.
Virtual CAN Network: High-Level Services for CAN Communication
Physical Time-Triggered Network: Stable and Validated Core Services

Fig. 1. Waist-Line Architecture with High-Level Services for CAN Communication

strong fault isolation, and consistent diagnosis of failing nodes through a membership service. The small number of core services eases a thorough validation (e.g., permitting a formal verification), which is crucial for preventing common mode failures as all high-level services and consequently all applications build on the core services.

The two high-level services depicted in Figure 1 (*CAN controller emulation* and *packet service*) realize the communication infrastructure for the CAN subsystem as a virtual network. The *virtual CAN network* is an overlay network that is established on top of the time-triggered physical network. This design leads to a separation of safety-critical functionality that needs to be certified to the highest criticality levels from those parts of the system that are specific to the non safety-critical applications (black area in Figure 1).

For the realization of the virtual CAN network, we employ the time-triggered physical network and perform a temporal subdivision of the communication resources. The media access control strategy of the time-triggered physical network is Time Division Multiple Access (TDMA). TDMA statically divides the channel capacity into a number of slots and controls access to the network solely by the progression of time. Each node is assigned a unique *node slot* that periodically reoccurs at a priori specified global points in time. A node sends messages during its node slot and receives messages during the node slots of other nodes. A sequence of node slots, which allows every node in an ensemble of n nodes to send exactly once, is called a TDMA round.

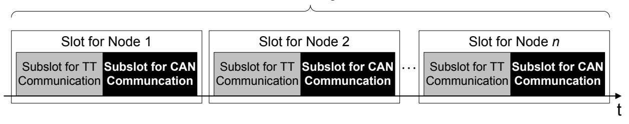

TDMA Round Consisting of *n* Node Slots
Slot for Node 1
Slot for Node 2
Slot for Node *n*
Subslot for TT Communication
Subslot for CAN Communcation
...
t

Fig. 2. Temporal Subdivision of TDMA Slots

Each node slot provided by the underlying time-triggered communication service is further subdivided into two subslots, namely a slot for the time-triggered communication and a slot for the event-triggered dissemination of CAN messages (see Figure 2). The assignment of the slots within a TDMA round to nodes, as well as the further subdivision into subslots for time-triggered and CAN communication is fixed at design time. This static allocation not only facilitates fault isolation between nodes, but also between CAN communication and time-triggered communication.

The *packet service* exploits the subslot for CAN communication in order to exchange CAN messages. The CAN message transmission requests of the CAN application software form a sequential stream of CAN messages. Inactivity messages are inserted during time intervals, in which applications do not produce messages. In every node, the packet service performs a fragmentation of outgoing messages into packets. These packets are placed in the node's CAN subslot. Furthermore, the packet service fuses packets received via the CAN subslots of other nodes into CAN messages and offers these messages to the application via incoming message queues. Depending on the size of the CAN subslot, it is possible for packets from multiple CAN messages to be placed into the CAN slot. Adversely, the packet service also allows a CAN subslot to be smaller than the size of a complete CAN message. In this case, the transmission of a CAN message is performed during multiple TDMA rounds.

The *CAN controller emulation* establishes the interface towards the application. For this purpose, this high-level service implements the register interface of commercial CAN controllers (e.g., i82527 [11], Philips SJA1000 [12]). Thereby, the CAN controller emulation permits to reuse CAN-based legacy application software that has been designed to directly access the respective CAN controller at the register level. In addition, the functionality of the CAN controller emulation comprises the queueing of incoming and outgoing messages, message filtering, and error detection.

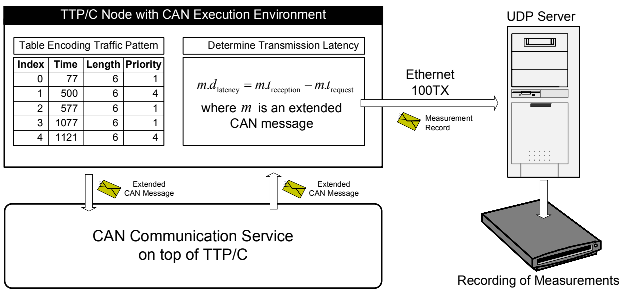
IV. Measurement and Simulation Framework

This section describes the framework that has been employed for measuring the transmission latencies in an implementation of virtual CAN networks and a simulation of a physical CAN network.

A. Measurement Framework

The measurement framework employs an implementation of virtual CAN networks in the TTA and employs the TTP/C protocol as the physical network. The nodes employed in the prototype implementation use the embedded real-time Linux variant Real-Time Application Interface (RTAI) [13] as their operating system. The RTAI version used for the prototype implementation combines a real-time hardware abstraction layer with a real-time application interface for making Linux suitable for hard real-time applications [14]. The packet service and the CAN controller emulation have been realized as middleware and implemented via Linux kernel modules.

A distributed measurement application uses off-line computed message transmission request tables as inputs for the virtual CAN network. The measurement application at every sender node accesses the virtual CAN network and requests message transmissions at the points in time specified in the request table (see Figure 3). The table also determines the length and identifier of each transmitted message. The data area of the CAN message contains a message index, which uniquely identifies a particular message transmission request along with the node from which the message originated.


TTP/C Node with CAN Execution Environment
Table Encoding Traffic Pattern
Determine Transmission Latency
$m.d_{\text{latency}} = m.t_{\text{reception}} - m.t_{\text{request}}$
where m is an extended CAN message
Ethernet 100TX
Measurement Record
UDP Server
Extended CAN Message
Extended CAN Message
CAN Communication Service on top of TTP/C
Recording of Measurements

Index	Time	Length	Priority
0	77	6	1
1	500	6	4
2	577	6	1
3	1077	6	1
4	1121	6	4

Fig. 3. Measurement Framework

Whenever a message m is received, we assign a timestamp $m.t_{\text{reception}}$ to the received message. Due to the availability of the global timebase of the TTA, we can compute the transmission latency $m.d_{\text{latency}}$ as follows:

$$m.d_{\text{latency}} = m.t_{\text{reception}} - m.t_{\text{request}} \tag{2}$$

where $m.t_{\text{request}}$ denotes the point in time at which the message needs to be sent according to the off-line computed message transmission request tables. The transmission latency $m.d_{\text{latency}}$ incorporates queuing delays of the virtual CAN network at the sender, latencies of the underlying network and execution times of the CAN middleware. A measurement record is constructed with the index of the sending TTP/C node, the sequence number of the message, the reception time $m.t_{\text{reception}}$, and the transmission latency $m.d_{\text{latency}}$. This measurement record is stored in a UDP packet and transferred to a workstation that executes a UDP server. The workstation collects the measurement records from TTP/C node and stores them into a dedicated file for each observing node for a later analysis.

B. Simulation Framework

We have employed a framework based on the MATLAB/Simulink toolbox TRUETIME for simulating the behavior of a physical CAN network, when provided with a particular message pattern as input. TRUETIME [15] is a MATLAB/Simulink-based simulation toolbox for real-time control systems. TRUETIME supports the simulation of the temporal behavior of tasks in a host computer, as well the simulation of the timing behavior of communication networks. For this purpose, it offers two Simulink blocks: a computer block and a network block. The blocks are variable-step, discrete, MATLAB S functions written in C++. The computer block S-function simulates a host computer with a flexible real-time kernel and user-defined tasks. Every task is associated with code (e.g., C functions), which is executed during the simulation. The network block operates event-driven and is triggered by messages entering or leaving the network. The network

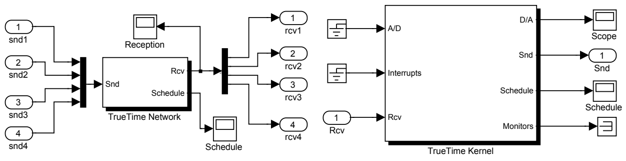
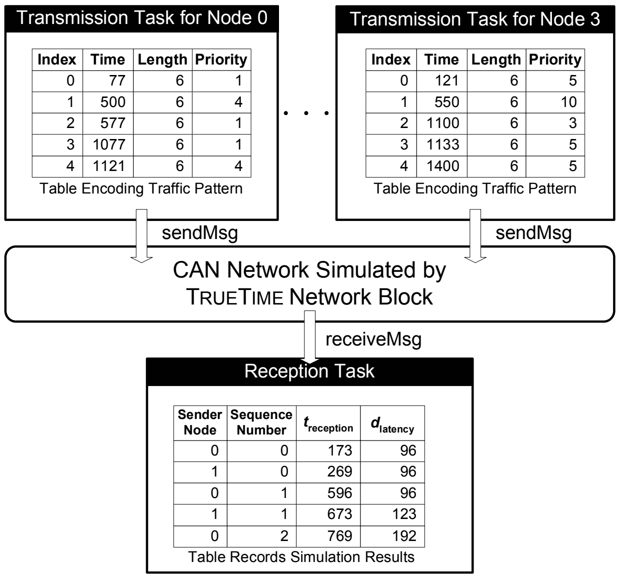


Fig. 4. MATLAB/Simulink Models for Node (left) and Network (right)

block is parameterized with the medium access control protocol (CSMA/CD, CSMA/CA, round robin, FDMA, or TDMA) and the transmission rate. A send queue is used to hold all messages currently enqueued in the network.

While TRUETIME has primarily been designed for analyzing the effects of timing non-determinism on control performance, the high flexibility of the computer block and the support for the CAN media access control protocol (CSMA/CA) make this tool well-suited for simulating a conventional CAN system. We use the TRUETIME computer and network blocks (see Figure 4) to model a CAN system consisting of four nodes. Every node employs a TRUETIME computer block that is connected to the network block modeling the CAN bus. In every node, a task is executed that transmits messages according to the message transmission request table as described in the measurement framework. At the points in time specified in this table, the task passes CAN messages to the TRUETIME network. Each CAN message is assigned the priority and length as specified in the table. The data area contains the point in time of the message transmission request and a unique index to identify the message.



Transmission Task for Node 0
Transmission Task for Node 3
. . .
sendMsg
sendMsg
CAN Network Simulated by TRUETIME Network Block
receiveMsg
Reception Task

Index	Time	Length	Priority
0	77	6	1
1	500	6	4
2	577	6	1
3	1077	6	1
4	1121	6	4

Table Encoding Traffic Pattern

Index	Time	Length	Priority
0	121	6	5
1	550	6	10
2	1100	6	3
3	1133	6	5
4	1400	6	5

Table Encoding Traffic Pattern

Sender Node	Sequence Number	$t_{reception}$	$d_{latency}$
0	0	173	96
1	0	269	96
0	1	596	96
1	1	673	123
0	2	769	192

Table Records Simulation Results

Fig. 5. Simulation Tasks

One of the nodes also hosts a reception task. The reception task is an interrupt handler that is triggered by message receptions. This task retrieves the CAN message from the TRUETIME network and calculates the transmission latency from the timestamp contained in the message and the current simulation time. The determined transmission latency is written into a file for a later analysis. The interplay between the transmission and reception tasks is visualized in Figure 5.

V. RESULTS

This section presents the measurement results from the implementation of virtual CAN networks and a simulation of a physical CAN network with message sets from a real-world application provided by the automotive industry. In addition, we have applied synthetic message sets in order to investigate the behavior of the virtual CAN networks under message loads exceeding 1 Mbps. For determining the bandwidth usage and points in time of message transmission requests, we use a message model that distinguishes between periodic, sporadic, and aperiodic messages.

At the underlying time-triggered communication schedule, we have used a communication schedule that provides CAN slots for four nodes with a TDMA round length of 320 μs. The CAN slot of each node has a size of 64 bytes.

A. Message Model

In our message model, the dissemination of a message m is repeatedly requested at points in time $\rho_{k,m} \in \mathbb{R}^+$ ($k \in \mathbb{N}$), where $\rho_{k,m}$ are stochastic variables. Every message m is characterized by two parameters, an interarrival time $d_m \in \mathbb{R}^+$ and random intervals $\delta_{k,m} \in \mathbb{R}^+$.

$$\forall k \in \mathbb{N}: \quad \rho_{k+1,m} - \rho_{k,m} = d_m + \delta_{k,m} \tag{3}$$

d_m specifies an a priori known minimum interval of time between two transmission requests of m. The stochastic variables $\delta_{k,m}$ cover the random part in the time interval between two transmission requests of m. For a particular message m, all stochastic variables $\delta_{k,m}$ possess the same distribution function δ_m, i.e. $\delta_{k,m} \sim \delta_m$.

The two message parameters d_m and δ_m employed in this message model allow to distinguish between three fundamental message types:

- A *periodic message* has a constant time interval between successive message transmission requests.

  $$m \text{ perodic } \leftrightarrow \ (\forall k \ \delta_{k,m} = 0) \ \wedge \ (d_m \in \mathbb{R} > 0)$$

- For a *sporadic message* the transmission request times are not known, but it is known that a minimum time interval exists between successive transmission requests.

  $$m \text{ sporadic } \leftrightarrow \ (\forall k \ \delta_{k,m} \sim \delta_m) \ \wedge \ (d_m \in \mathbb{R} > 0)$$

- For an *aperiodic message* neither the message transmission request times are known nor a minimum time interval between successive transmission requests.

  $$m \text{ aperiodic } \leftrightarrow \ (\forall k \ \delta_{k,m} \sim \delta_m) \ \wedge \ (d_m = 0)$$

B. Automotive Message Set

The message sets from the real-world automotive application originate from a powertrain network with a bandwidth of 500 kbps and consist of 102 periodic messages.

The message periods range from 3.3 ms to 1 s, the number of data bytes is between 2 and 8 bytes. The network utilization is 60%, i.e. the overall network bandwidth required for the exchange of these messages is 300 kbps. Messages use the standard CAN format and contain 47 control bits, thereby resulting in a total message size between 47 and 111 bits.

The message sets from the automotive industry were used as inputs for both the simulation of a physical CAN network and the implementation of a virtual CAN network. In the simulation, we have modeled a CAN network with a bandwidth of 500 kbps.

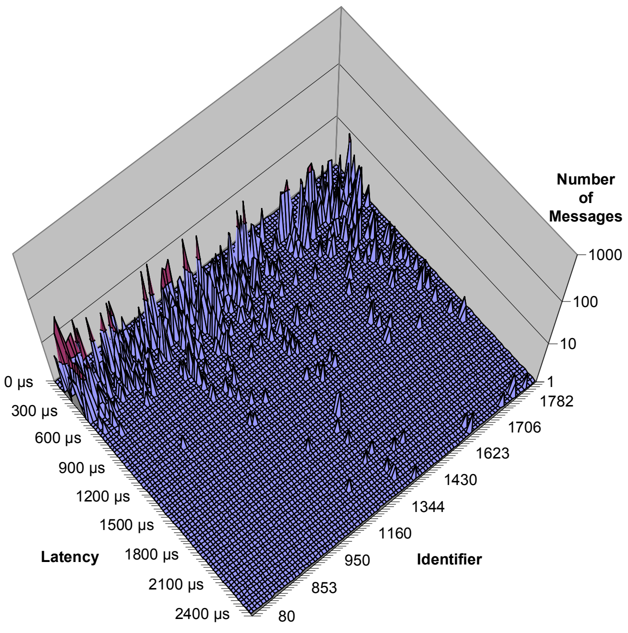


Fig. 6. Simulation Results for Automotive Message Sets (Logarithmic Scaling of z-axis)

The simulation results for the automotive message sets are depicted in Figure 6. The x-axis represents the message transmission latencies. The message identifiers are distinguished along the y-axis. These identifiers range from 80 to 2004 and denote the message priority. Larger CAN identifiers correspond to lower message priorities. The distance along the z-axis represents the number of messages with a given message priority and transmission latency. Figure 6 shows that high priority messages make up for a large amount of the overall bandwidth. The third of messages with the highest priority makes up for 49% of exchanged messages. The simulation results also demonstrate the high average performance of CAN. 97% of all message transmissions possess transmission latencies below 1 ms. However, transmission latencies vary considerably. The logarithmic scaling of the z-axis in Figure 6 emphasizes the rare cases, in which transmission latencies significantly differ from the average values.

Figure 7 depicts the measured transmission latencies for the automotive message sets in the implementation of the virtual CAN network. The observed worst-case latency is 608 μs, i.e. significantly lower compared to the 3165 μs of the physical CAN network. The best-case latency is 147 μs. The average message transmission latency of the measurements is 323.51 μs. An overview of the latencies observed in the simulation and measurements is contained in Table I.

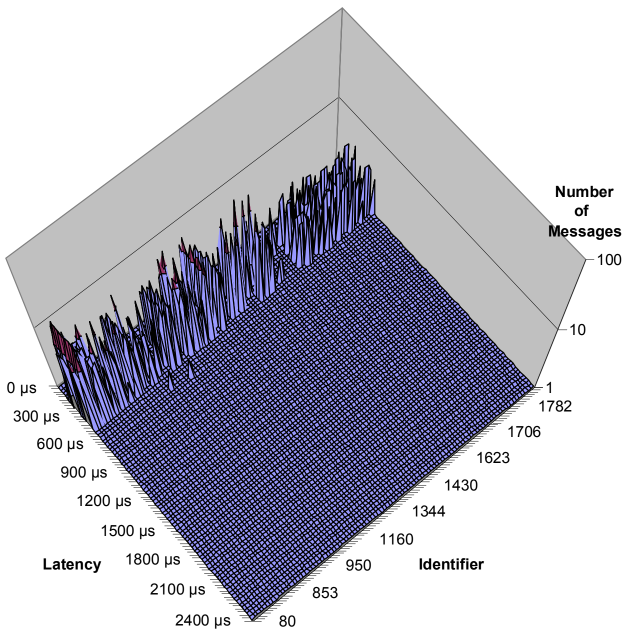


Fig. 7. Measurement Results for Automotive Message Sets (Logarithmic Scaling of z-axis)

	Min	Max	Mean	Variance
Physical CAN	126 μs	3165 μs	350.11 μs	78581.45
Virtual CAN	147 μs	608 μs	323.51 μs	8371.96

TABLE I
LATENCIES IN PHYSICAL AND VIRTUAL CAN NETWORKS

C. Synthetic Message Set

The synthetic message set consists of CAN messages complying with the extended format [2]. An extended CAN message contains 67 control bits and between 0 and 8 bytes of effective data. Consequently, a complete CAN message has a length between 67 and 131 bits.

We have employed two groups of input message sets. One group consists of sporadic messages, a second one uses aperiodic messages. In both groups, we have specified maximum bandwidth values in order to determine the effects of different message loads on the communication system. We have used the following expression for the determination of the bandwidth usage of a message m:

$$bandwidth(m) = \frac{b_m}{d_m + \mathbb{E}\left[\delta_m\right]} \tag{4}$$

where d_m denotes the minimum interarrival time of transmission requests of message m. δ_m is the distribution function of the stochastic variables $\delta_{k,m}$ that represent the random intervals between transmission requests. b_m is the length of message m in bits. For our synthetic message sets $b_m = 99$ bits, due to the use of extended format CAN messages with 4 data bytes.

We use the assumption of a uniform distribution for the lengths of random intervals $\delta_{k,m}$ of message transmission requests. For a particular message m, we consider all random intervals $\delta_{k,m}$ to possess the same distribution function $U(0, u_m)$.

$$\delta_{k,m} \sim \delta_m = U(0, u_m) \quad (5)$$

This leads to the following sum for the bandwidth usage of a message set M:

$$bandwidth(M) = \sum_{m \in M} \frac{99}{d_m + 0.5 \cdot u_m} \quad (6)$$

Based on this formula for the calculation of the bandwidth usage of a message set, we have constructed message sets with sporadic and aperiodic messages, a bandwidth usage 4 Mbps, and a fixed number of 80 messages. For the aperiodic message sets, we have started with an upper bound $u_m = 1\,\text{ms}$ for the random interval length and added a linearly increasing summand until reaching the bandwidth limit with the predefined number of 80 messages. For the sporadic messages sets, the random interval length starts at $u_m = 500\,\mu\text{s}$, the initial minimum interarrival time d_m is $250\,\mu\text{s}$. Both the random interval length and the minimum interarrival time of subsequent messages are increased with a linearly increasing summand until reaching the bandwidth limit with the predefined number of messages.

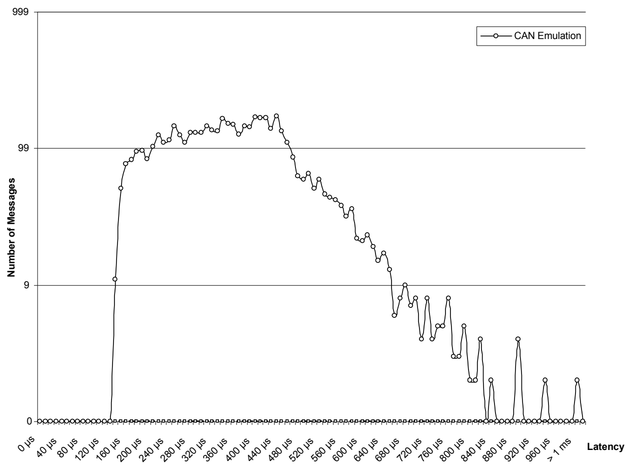


Fig. 8. Measurement and Simulation Results (Aperiodic Message Set, 4 Mbps): *Message transmission latencies vary between 147 µs and 1.1 ms at the overall bandwidth consumption of 4 Mbps by CAN messages.*

Figure 8 depicts the measurement results for the aperiodic message sets with an overall bandwidth consumption of 4 Mbps. The results for the sporadic message are shown in Figure 9. Since a physical CAN network supports a maximum bandwidth of 1 Mbps, we provide only measurement results from the virtual CAN network.

The figure contains a diagram with the distributions of the observed message transmission latencies. The transmission latencies are distinguished along the x-axis. The y-axis represents the number of messages observed with a particular transmission latency. We have chosen a logarithmic scaling of the y-axis in order to emphasize rare latency values in the diagrams.

These results demonstrate the ability of virtual CAN

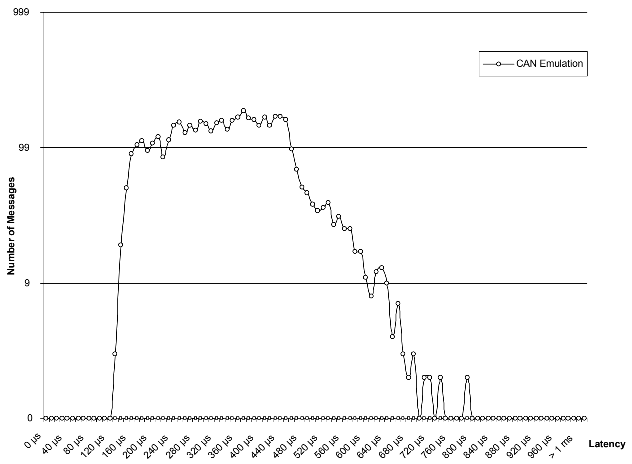


Fig. 9. Measurement and Simulation Results (Sporadic Message Set, 4 Mbps): *Message transmission latencies vary between 147 µs and 796 µs.*

networks to handle bursty aperiodic and sporadic message sets with bandwidths above 1 Mbps. During the measurements, we have observed a maximum latency of 1.1 ms for the aperiodic 4 Mbps message set. For the sporadic message set, the message transmission latencies vary between 147 µs and 796 µs. For both message sets, the CAN message with the highest priority has a maximum transmission latency of 479 µs.

VI. Discussion of Measurement Results

The simulation and measurement results demonstrate the ability of virtual CAN networks to integrate CAN-based legacy applications. Virtual CAN and physical CAN exhibit similar transmission latencies. The physical CAN network excels with respect to the maximum latency for the highest priority message by only 35 µs. The communication jitter of the physical CAN network, on the other hand, is more than six times as large as the communication jitter of the virtual CAN network. In addition, the virtual CAN network supports future CAN applications with bandwidth requirements of more than 1 Mbps.

A. Communication Latencies

In Section II, we have described the latencies of a physical CAN network. In the absence of faults, the shortest transmission latency can be guaranteed for the highest priority message h. Equation 1 for the maximum transmission latency of h contains the time C_h for transmitting message h, as well as the maximum transmission duration of a lower priority message. Since the automotive message set originates from a 500 kbps CAN network and the highest priority CAN message h uses the standard format and possesses 4 data bytes, C_h can be calculated as follows:

$$C_h = \left(\left\lfloor \frac{34 + 8 \cdot 4}{5} \right\rfloor + 47 + 8 \cdot 4 \right) \cdot 2 = 184\,\mu\text{s} \quad (7)$$

The maximum length of data bytes for a lower priority message is 8. Hence, the second summand is:

$$\max_{\forall k \in lp(C_h)} (C_k) = \left(\left\lfloor \frac{34 + 8 \cdot 8}{5} \right\rfloor + 47 + 8 \cdot 8 \right) \cdot 2 = 260\,\mu s \quad (8)$$

If we abstract from the queuing jitter, we can establish the bound $R_h = 444\,\mu s$ for the maximum transmission latency of the highest priority message h. This latency also fits to the measurement results from the simulation of the physical CAN network.

In the latency measurements for the virtual CAN network we have observed a maximum latency of 479 μs for the highest priority message. While this bound would be sufficient for providing the maximum guaranteed latency of a 250 kbps CAN network, this value is slightly larger than the bound of the 4 data byte message at the 500 kbps CAN network. However, we assume that the difference in the maximum latency of 35 μs is no problem for the integration of legacy applications.

In order to explain the maximum measured latency of the virtual CAN network, we look at two determining parameters: the communication schedule and the processing overhead of the operating system and middleware services.

- **Communication Schedule.** Every node sends exactly once in a TDMA round with a duration of 320 μs in the schedule employed for the measurements. The TDMA round duration directly implicates the maximum message transmission latency. Transmission requests are not phase aligned with the underlying TDMA scheme, but can occur at arbitrary points in time within a round. Hence, in the worst case a request occurs immediately after the start of a node's slot. In this case, a message is delayed for a complete TDMA round (i.e. 320 μs) until the node's slot reoccurs.
- **Processing Overhead.** The processing overhead comprises the delay introduced by the operating system and middleware, such as the interrupt latency for time-triggered activation of the CAN middleware, as well as the execution of the middleware code. In the prototype implementation, this overhead is approximately 80 μs and occurs at both the sender and at the receiver.

Therefore, a further reduction of the maximum latency for the virtual CAN network can be reached by either decreasing the TDMA round length or by optimizing the CAN middleware code in order to reduce the run-time overhead of the middleware service. The CAN prototype implementation has been designed to accommodate different time-triggered communication schedules in order to support the first modification. For example, a shorter TDMA round length would permit a reduction of the round length, thereby allowing the provision of latency guarantees in equivalence to physical CAN networks with bandwidths of 500 kbps or 1 Mbps.

Consequently, the key element for establishing a certain bound for the communication latencies of the virtual CAN network is the layout of the time-triggered communication schedule. Virtual CAN communication latencies can be controlled via the TDMA round lengths and the number of bytes dedicated to the virtual CAN communication. In general, relaxed requirements with respect to transmission latency guarantees allow to increase round lengths or to decrease the number of bytes dedicated to the virtual CAN network.

If the reproduction of transmission latencies in the presence of faults is sufficient, the duration of inaccessibility times determines the acceptable transmission latencies. The maximum latency of 479 μs observed during the measurements of the virtual CAN network is significantly below the possible inaccessibility time of 2.2 ms in a 1 Mbps CAN network. In the prototype implementation, the fault-tolerant TTP/C protocol [16] ensures that faults covered by the fault hypothesis of the TTA [17] have no impact on the temporal behavior of message exchanges between correct nodes.

B. *Communication Jitter*

While control algorithms can be designed to compensate a known delay, communication jitter (i.e. the difference between the maximum and minimum value of the communication delay) brings an additional uncertainty into a control loop that has an adverse effect on the quality of control [18].

The simulation and measurement results demonstrate that the communication jitter of the physical CAN network is significantly larger than the communication jitter of the virtual CAN network. While the communication jitter of the physical CAN network is 3 ms, the virtual CAN network has a jitter of only 461 μs.

The reasons for this difference in the communication jitter is the higher bandwidth provided by the virtual CAN network, as well as a different strategy for the multiplexing of bandwidth. The physical CAN network shares bandwidth system wide. Thus, the transmission latency of a message at a particular node depends on the transmission requests at the other nodes in the system. In the virtual CAN network, on the other hand, bandwidth of a node is shared only locally between the different message produced by the application software at this node. Hence, the message transmission latency of messages at different nodes are independent.

C. *Exceeding of CAN Limitations*

The arbitration mechanism of CAN requires bits to stabilize on the channel and limits the maximum bandwidth to 1 Mbps at a network length of 40 m. The bit length must be at least as large as the propagation delay.

The measurement results in Section V demonstrate the ability of the virtual CAN network to handle bandwidths that exceed 1 Mbps. For the 4 Mbps message sets, a maximum transmission latency of 1.1 ms has been observed.

VII. Conclusion

The comparison of the temporal performance of virtual and physical CAN networks performed in this paper provides a basis for designers who intend to reuse

CAN-based application despite the migration to a time-triggered platform. Such a migration can be useful, if the introduction of safety-critical functionality (e.g., steer-by-wire and brake-by-wire in the automotive industry) demands for a time-triggered communication service with high bandwidth, fault-tolerance and deterministic behavior. Virtual CAN networks eliminate the need to provide separate physical networks for safety-critical functions and already existing CAN-based applications. As overlay networks on the time-triggered communication service, these virtual CAN networks enable significant cost and reliability benefits through the reduction of connectors and wiring.

In addition to supporting legacy applications, virtual CAN networks can also handle the communication requirements of future CAN-based applications with bandwidth requirements beyond 1 Mbps. These applications can also benefit from the availability of additional services (e.g., clock synchronization provided by the time-triggered communication protocol) and the lower jitter compared to a physical CAN network.

Acknowledgments

This work has been supported in part by the European IST project DECOS (IST-511764) and the European IST project ARTIST2 (IST-004527).

References

[1] R. Obermaisser, "CAN Emulation in a Time-Triggered Environment," in *Proceedings of the 2002 IEEE International Symposium on Industrial Electronics (ISIE)*, vol. 1, 2002, pp. 270–275.

[2] *CAN Specification, Version 2.0*, Robert Bosch Gmbh, Stuttgart, Germany, 1991.

[3] K. Tindell and A. Burns, "Guaranteed message latencies for distributed safety-critical hard real-time control networks," Dept. of Computer Science, University of York, Tech. Rep. YCS229, June 1994.

[4] P. Veríssimo, J. Rufino, and L. Rodrigues, "Enforcing real-time behaviour of LAN-based protocols. In Proceedings of the 10th IFAC Workshop on Distributed Computer Control Systems, Semmering, Austria, IFAC," September 1991.

[5] J. Rufino and P. Veríssimo, "A study on the inaccessibility characteristics of the controller area network," in *2nd International CAN Conference*, London, United Kingdom, Oct. 1995.

[6] R. Obermaisser, *Event-Triggered and Time-Triggered Control Paradigms – An Integrated Architecture*, ser. Real-Time Systems Series. Kluwer Academic Publishers, Nov. 2004.

[7] *DO-178B: Software Considerations in Airborne Systems and Equipment Certification*, Radio Technical Commission for Aeronautics, Inc. (RTCA), Washington, DC, Dec. 1992.

[8] *IEC 61508-7: Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems – Part 7: Overview of Techniques and Measures*, IEC: International Electrotechnical Commission, 1999.

[9] E. Bretz, "By-wire cars turn the corner," *IEEE Spectrum*, pp. 68–73, Apr. 2001.

[10] J. Rushby, "Bus architectures for safety-critical embedded systems," in *Proceedings of the First Workshop on Embedded Software (EMSOFT 2001)*, ser. Lecture Notes in Computer Science, T. Henzinger and C. Kirsch, Eds., vol. 2211. Lake Tahoe, CA: Springer-Verlag, Oct. 2001, pp. 306–323.

[11] *82527 Serial Communications Controller, Controller Area Network Protocol*, Intel Corporation, Dec. 1995.

[12] *SJA1000 Stand-alone CAN controller Product Specification*, Philips Semiconductors, Eindhoven, Jan. 2000, www-us.semiconductors.philips.com/pip/SJA1000.html.

[13] D. Beal, E. Bianchi, L. Dozio, S. Hughes, P. Mantegazza, and S. Papacharalambous, "RTAI: Real-Time Application Interface," *Linux Journal*, April 2000.

[14] "RTAI Programming Guide, Version 1.0," Dipartimento di Ingegneria Aerospaziale Politecnico di Milano (DIAPM), Italy, September 2000, available at http://www.rtai.org.

[15] D. Henriksson, A. Cervin, and K. Arzen, "Truetime: Simulation of control loops under shared computer resources," in *Proceedings of the 15th IFAC World Congress on Automatic Control*. Barcelona, Spain: Department of Automatic Control, Lund Institute of Technology, July 2002.

[16] H. Kopetz, *Specification of the TTP/C Protocol*. Schönbrunner Straße 7, A-1040 Vienna: TTTech, July 1999, available at http://www.ttpforum.org.

[17] H. Kopetz and G. Bauer, "The time-triggered architecture," *IEEE Special Issue on Modeling and Design of Embedded Software*, Jan. 2003.

[18] H. Kopetz, *Real-Time Systems, Design Principles for Distributed Embedded Applications*. Boston, Dordrecht, London: Kluwer Academic Publishers, 1997.