A Network Virtualization Approach for Performance Isolation in Controller Area Network (CAN)

Christian Herber, Andre Richter, Thomas Wild, Andreas Herkersdorf
Technische Universität München - Institute for Integrated Systems
Munich, Germany
{christian.herber, andre.richter, thomas.wild, herkersdorf}@tum.de

***Abstract*—An important trend in automotive CPS is the shift from federated to integrated IT architectures, where multiple functions are consolidated on shared electronic resources instead of distributed electronic control units (ECUs). It is driven by increasing complexity, cost and installation space requirements of todays architectures. However, side-by-side integration of mixed-criticality functions poses new challenges with respect to safety and security.**

To achieve isolated performance for multiple integrated partitions with different criticalities, an efficient separation within computing and communication resources is required. This paper introduces a network virtualization approach for CAN, which enables concurrence of mixed-criticality communication on a single physical CAN bus through a strict performance isolation. We present a design concept as well as a prototypical implementation. The feasibility of our approach is demonstrated by an analytic evaluation of message latencies and through experimental case studies.

***Index Terms*—Controller area network, CAN, network virtualization, automotive electronics, embeddded virtualization, mixed-criticality.**

I. Introduction

Automotive IT architectures have evolved into complex cyber-physical systems (CPS). In modern premium vehicles around 70 electronic control units (ECUs) communicate through a number of specialized fieldbuses like e.g. Controller Area Network (CAN), FlexRay and MOST with more than 2.7 km in wire length [1]. In these architecture multiple functional domains (e.g. powertrain, chassis, infotainment) coexist while having different requirements with respect to real-time capability, security, safety, and dynamics. This trend will increase in future cars with the introduction of Car2X connectivity and automotive cloud computing.

To handle the complexity of these mixed-criticality systems, fieldbuses are usually exclusive to specific domains and each electronic function has its own ECU. This approach has led to a good fault isolation and eases the process of qualification, but makes inefficient use of computational and communication resources. Because the installation space demand and cost are limiting the scalability of current architectures, new approaches are needed to cope with the requirements of future automotive CPS.

This work was funded within the project ARAMiS by the German Federal Ministry for Education and Research with the funding IDs 01|S11035. The responsibility for the content remains with the authors.

Automotive OEMs are planning to consolidate multiple ECUs on multi-core controllers in so called domain controlled architectures [2]. Here, the inherent parallelism of multi-core processors is used for concurrent execution of multiple electronic functions on a shared platform. This centralization of functions reduces the number of ECUs in a car, increases resource utilization and provides a tighter coupling of functions, but also introduces challenges regarding performance and fault isolation [3].

By integrating electronic functions, error sources and attack vectors are integrated as well. Safe and secure side-by-side execution of electronic functions on a shared multi-core platform can be enabled through virtualization [4], [5]. Here, isolated computational resources are allocated to multiple partitions or virtual machines (VMs), which allows performance guarantees and fault isolation among the VMs.

When computation nodes are interconnected as in automotive CPS, additional challenges arise. On the one hand, efficient and scalable implementations of centralized architectures require communication resources to be shared, but on the other hand, safe communication requires isolation among mixed-criticality communication flows.

Because current fieldbuses like CAN do not provide sufficient performance isolation, centralized architectures are reliant on dedicated physical buses for functional domains of different criticality. To overcome this design constraint, fieldbuses have to be adapted to be used within different functional domains communicating on a shared physical medium, without drawbacks in safety or security.

In this paper, we present a network virtualization approach for Controller Area Network that provides isolated communication resources in the form of abstracted, virtual Controller Area Networks (VCANs). Reserved bandwidths are guaranteed by a decentralized admission control scheme, which additionally ensures small temporal interference and fault isolation between concurrent VCANs. The approach enables mixed-criticality scenarios, in which e.g. powertrain ECUs with strict real-time requirements and high qualification effort can share a common bus with infotainment ECUs, which produce dynamic best effort traffic and are rather constraint by bandwidth requirements and security aspects.

The remainder of this paper is structured as follows: Section II presents related work regarding network virtualization and mixed-criticality systems. In Section III, the fundamentals for CAN are introduced. Based on this, a concept for network virtualization in CAN is presented in Section IV. Additionally, a bound for the delay of communication in a VCAN is presented. Using the real-time analysis of message latencies in VCANs introduced in Section V, we evaluate the timing behavior of our approach in Section VI. Additional timing information is derived through an experimental approach in Section VII. In Section VIII, we describe an architectural implementation as well the hardware overhead necessary for our extensions and finally, Section IX concludes this paper.

II. Related Work

Mixed-criticality systems consolidate functions that differ greatly in real-time, safety, reliability and security requirements on shared resources. Because the integration of such functions requires the whole system to be certified at the highest criticality level if interference between such functions is possible, isolation mechanisms for computation and communication resources are required. In [6], Pellizoni et al. propose a design methodology for SoC architectures, which achieves strong isolation among different criticalities through static resource reservations. A NoC for mixed-criticality systems, which is capable of providing isolated real-time guarantees in the presence of best effort communication, is presented in [7].

Other approaches address mixed-criticality systems by deriving schedules and schedulability analyses that take advantage of criticality dependent parameters in such systems to derive optimized schedules. Baruah et al. [8] exploit the fact that worst-case execution times (WCETs) of tasks are dependent on certification requirements associated with different criticalities. Based on this assumption, an improved scheduling for mixed-criticality systems on a preemptive uniprocessor platform is derived. In [9], Burns and Davis propose a mixed-criticality protocol for CAN. Here, different criticality modes are introduced, in which the cycle time of a critical message is dependent on the current mode. Isolation is achieved through a trusted network component that constrains messages to conform to their specification.

Network virtualization is a concept that is most prominently used in Ethernet and IP-based networks to allow multiple logical networks to coexist on shared physical network resources. It enables concurrent networks, which are abstractions of the underlying physical networks using a subset of the nodes and edges available in the network. The concepts are used to improve the isolation and security of sub-networks, increase the overall utilization and to reduce complexity [10].

While such networks do not require to meet latency requirements as stringently as in automotive environments, a number of approaches have been introduced that aim at providing bandwidth guarantees in virtualized networking environments. In [11], a data center virtualization approach is presented, in which a logically centralized manager ensures bandwidth guarantees in virtual networks interconnecting virtual servers. A similar, but decentralized mechanism is introduced in [12], which enforces bandwidth guarantees for distributed cloud services based on distributed, cooperating policers.

The concept of network virtualization has been applied to many other networks, ranging from mobile communication networks like LTE [13] to on chip interconnects like NoCs [14], [15]. No research has yet been conducted regarding the virtualization of embedded fieldbuses. Previous research for virtualization in transportation CPS has rather dealt with platform virtualization [4] and I/O virtualization [16], [17].

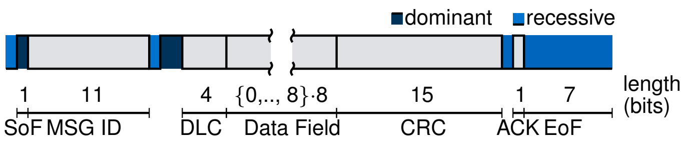

Fig. 1. Standard CAN Data Frame

Here, we introduce the concept of a virtual Controller Area Network (VCAN), which allows mixed-criticality components to transfer messages through isolated communication networks that share a single physical CAN. In contrast to Ethernet/IP based network virtualization approaches, this work focuses strongly on isolated real-time capabilities of the virtual networks. While previous work regarding mixed-criticality in CAN [9] has focused on developing mixed-criticality message schedules, we provide a performance isolation mechanism, which allows message latencies to be guaranteed without knowledge of the communication in other VCANs. This reduces the systems complexity and eases the burden of certification and qualification.

III. Controller Area Network

Controller Area Network (CAN) is a serial field bus with a bandwidth of up to 1 Mbit/s. It is the most widely used bus in automotive electronics being employed in many functional domains like powertrain, body, safety, infotainment and comfort. It was designed to provide safe and cost-efficient communication between ECUs in a car.

The nodes within a CAN do not have addresses and all messages are broadcasted. The content of a message is derived from its message ID. This ID is also used for the arbitration of messages on the CAN bus. A standard CAN data frame is depicted in Fig. 1, in which logical high and low levels are depicted by dominant and recessive states.

Access towards the CAN bus is managed using a non-preemptive, strict priority scheme, where lower message IDs guarantee a higher priority. This arbitration process is handled bitwise by every node on the bus. All nodes that have a frame ready for transmission start transmitting the message IDs concurrently. Dominant bits override recessive ones, and nodes that have been overwritten back off the arbitration and only one participant remains in the end.

CAN is an asynchronous protocol and does not require nodes to have a common time base. However, the sampling points of nodes have to be synchronous during the transmission of a frame. A soft synchronization is done after each transition from recessive to dominant state on the bus. After five bits of equal state, a bit with opposite level is inserted to enable this synchronization (bitstuffing). After a longer idle phase, synchronization can be lost and will be restored upon the transmission of the Start-of-Frame (SoF) bit (hard synchronization).

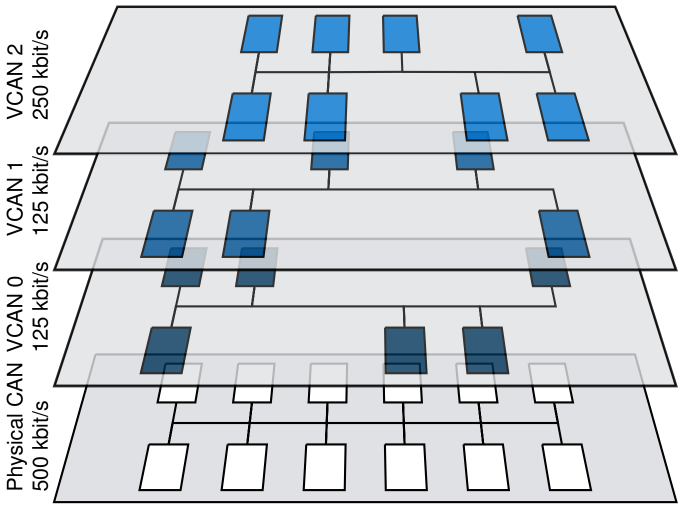

Fig. 2. Virtual Controller Area Networks (VCANs) coexisting on a shared physical CAN bus.

IV. Design of a virtualized CAN

The goal of this work is to enable multiple concurrent virtual CANs (VCANs) to coexist on a shared physical bus in mixed criticality scenarios. Each VCAN must have its exclusive namespace and guaranteed performance. Within CAN, the main performance metrics are the bandwidth and the latencies experienced by each message. While CAN is used for many real-time critical control applications, it is also used for best effort traffic, which is mainly constrained by bandwidth requirements. For VCANs that rely on real-time communication, the latencies should be as close as possible to the native case with an exclusive CAN bus of equivalent bandwidth. The goals have to be achieved under the following design space constraints:

1) The CAN protocol must not be modified. Any changes necessary should happen within the participating nodes.
2) The changes within the nodes should be as minimal as possible.
3) The admission control should be realized in a decentralized way. CAN does not have a central arbiter and the introduction of one would add a high overhead.
4) A high utilization of the physical communication resources is required. The available bandwidth should be fully utilizable by the VCANs.

In the following, we present concepts for the two main design challenges: the naming of messages within different VCANs and the admission control, which enforces bandwidth guarantees and minimizes temporal interference. Afterwards, we show how the admission control mechanism can ensure fault isolation among VCANs.

A. Naming

Naming in CAN is solely based on message IDs. Within a single CAN, the message IDs have to be unique in order to avoid collisions on the bus and to have an explicit relation between ID and message content. In addition, this ID also specifies a strict priority on the CAN bus.

Under these conditions, an applicable scheme has to be found, which allows the IDs within each VCAN to coexist on the physical bus without inference. The easiest way of dividing the ID space among VCANs is the use of tags within the message IDs, from which VCAN ID of a message can be derived.

Placing a VCAN tag within the ID of a message decreases the number of total IDs available. Assuming 4 bits to be reserved for VCAN tagging (support for 16 VCANs), there are still 7 bits left per VCAN for message identifiers (support for 128 messages). While this should be sufficient in most realistic cases, scenarios in which single VCANs have a high demand for IDs might not be possible to realize in this way. In such scenarios, encoding the VCAN identifiers with variable length (e.g. by Huffman Coding) could resolve this issue.

An important question is the placement of the VCAN tag within the ID, because this directly influences the priority of messages. This effect might be wanted or not, depending the application scenario and admission control scheme applied. We decided to place the tag within the most significant bits of the message ID. Therefore, the CAN bus arbitration enforces strict priorities among the VCANs, where VCAN $v = 0$ has the highest priority. Such a strict priority scheme is only capable of giving isolated performance guarantees for the highest priority VCAN, and therefore has to be extended by an additional admission control scheme.

B. Admission Control

In order to achieve performance isolation among different VCANs, the transmission of messages has to be restricted by an admission control mechanism. For the design of the admission control, we assume that the VCANs are spatially isolated by placing a VCAN tag at the most significant bits of the message ID. The physically available bandwidth r_{phy} has to be divided among the VCANs. For a VCAN v with a reserved bandwidth of r_v, the latencies experienced should be similar to the ones on a physical CAN bus with the same bandwidth.

Dividing the bandwidth of the physical bus towards VCANs can be achieved by means of time division multiplex (TDM) methods, which ensure that frames from different VCANs can access the CAN bus without exceeding a reserved bandwidth. A variety of methods for TDM methods are known, including time division multiple access (TDMA), in which slots of a fixed length are served in a deterministic pattern. However, such schemes (used e.g. in Time-Triggered CAN (TTCAN) [18]) require a global synchronization and introduce additional overhead if packet/frames do not all have the same length.

The complexity that comes along with such time-triggered schemes can be overcome by using an asynchronous scheme for admission control. Statistical TDM (STDM) is not based on strict timing checks, but rather enforces bandwidth guarantees by policing and/or traffic shaping based on statistical measures. Such a scheme can be implemented using admission control based on a token bucket or leaky bucket concept. These mechanism have been used e.g. in ATM for virtual channel arbitration [19].

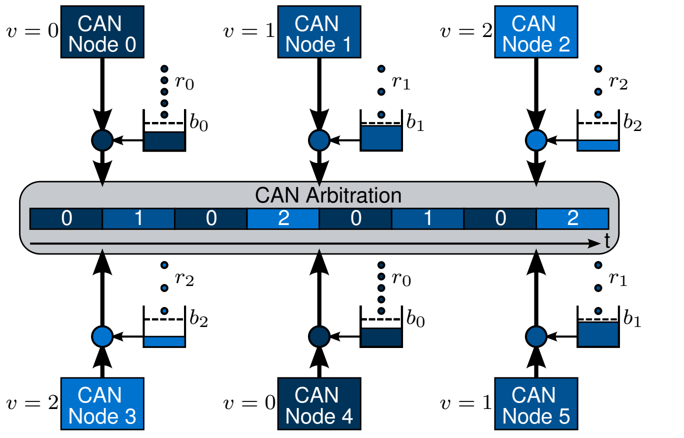

Fig. 3. Admission control towards the CAN bus is handled by token buckets for each VCAN, which enforce the reserved bandwidth. The fill level within each node of a VCAN is consistent.

The principle of token bucket policing can be described as an analogy to a bucket, in which tokens are added at a fixed rate proportional to the required bandwidth, and are removed upon transmission proportional to the size of the frame. It is characterized by its size b_v and the rate at which tokens are added r_v, which is equivalent to the reserved bandwidth.

Logically, the admission control is realized by constraining the transmission of frames by one token bucket policer per VCAN. In practice, this policing has to be performed decentralized and consistently within each node as depicted in Fig. 3. The highest priority message in every node is eligible to participate in the CAN bus arbitration, if sufficient tokens are available. If no higher priority message from the same VCAN and no message from a higher priority VCAN are participating in the arbitration, the message can be transmitted and the tokens will be removed according to its transmission time.

Because the admission control is decentralized, only information available to all nodes should be used for the policing. Due to the broadcasting behavior of CAN, any information that is propagated on the CAN bus can be seen by every node, and is therefore fit to be used for admission control. Other information, like the priority or length of a message buffered locally within a node may not be used. Therefore, a transmission should only be possible, if enough tokens for the transmission of the maximum length message are available. Otherwise, priority inversions could occur, where a short, low priority message is transmitted despite a longer, higher priority message being queued in a different node. The token bucket fill level $fl_{tx,v}$, at which any message is eligible for transmission can be derived as

$$fl_{tx,v} = \lceil C_{max,v} \cdot (r_{phy} - r_v) \rceil \,, \tag{1}$$

where $C_{max,v}$ the worst-case transmission time on the bus of the longest message within VCAN v and $r_{phy} - r_v$ is the rate at which tokens are reduced during transmission.

The consistency of the bucket fill levels is important for proper operation. Because the sample points of the CAN nodes are constantly synchronized during transmission, these points can be used to add tokens at a consistent rate and to remove tokens according to the actual transmission time of a message including bit stuffing. However, when the bus is idle, oscillator quality may cause a drift of the different nodes. This drift can be avoided, when idle phases are filled with dummy transmissions, where a designated node is sending minimum length messages of lowest priority. These messages are not accounted for any VCAN and have no negative impact on the worst-case timing of the overall system.

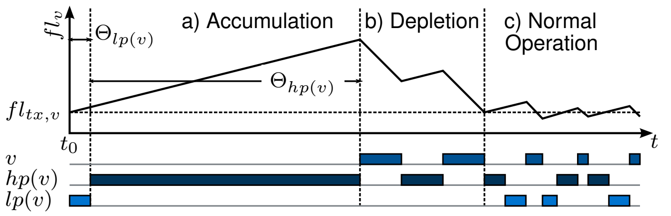

Fig. 4. Fill level of VCAN v during a burst scenario by higher priority VCANs.

C. VCAN Delay and Token Bucket Size

Two parameters that are closely coupled are the size of the token buckets b_v and the maximum delay Θ_v experienced in VCAN communication flows. Larger token buckets allow larger bursts and can therefore cause delays in lower priority VCANs. However, a minimum bucket size is necessary to prevent a bucket overrun during a maximum length delay, which would result in a loss of bandwidth.

The minimum token bucket size capable of maintaining bandwidth guarantees during phases of maximum delay is

$$b_v = fl_{tx,v} + \lceil \Theta_v r_v \rceil. \tag{2}$$

It equals the sum of the minimum fill level necessary for transmission as well as tokens added during Θ_v, the duration of a maximum sized burst of messages by all VCANs with higher priority than v.

In the following, we will present the worst-case scenario, which maximizes the delay for any VCAN v. We then give an upper bound for Θ_v, which can be used to determine the bucket sizes based on (2). Throughout the scenario, higher priority VCANs minimize the possible communication flow in VCAN v by transmitting at the maximum allowed rate. The scenario is divided into three different phases, which are illustrated in Fig. 4:

a) Accumulation: A message from a lower priority VCAN of maximum length $C_{max,lp(v)}$ has started transmission at $t = t_0$. With $\mathcal{V}$ being the set of all VCANs, the set of lower priority VCANs is defined as $lp(v) = \{u \in \mathcal{V} : prio(u) < prio(v)\}$. Afterwards, higher priority VCANs occupy the bus until their buckets are empty and no more transmissions are possible. VCAN v becomes able to transmit $(fl_v = fl_{tx,v})$ an infinitesimal time after t_0 and is delayed during the interval $(t_0, t_0 + \Theta_v]$.

b) Depletion: At the beginning of this phase, higher priority VCANs are not allowed to transmit, because they consumed their tokens. VCAN v has accumulated tokens equivalent to its bucket size. During this phase, the accumulated tokens allow VCAN v to transmit above its reserved bandwidth, which leads to a depletion of the tokens until no more transmission is possible.

c) Normal Operation: After the depletion of tokens, the transmission of VCAN v and higher priority VCANs is restricted to their reserved bandwidth r_v. This implies that lower priority VCANs are allowed to transmit. The tokens accumulated at the end of phase a) can only be reached again, if the initial conditions are again established and the scenario is repeated.

The longest VCAN delay occurs during the accumulation phase and is composed of blocking from higher and lower priority VCANs.

$$\Theta_v = \Theta_{lp(v)} + \Theta_{hp(v)}, \tag{3}$$

where $\Theta_{lp(v)} = C_{max,lp(v)}$ and $\Theta_{hp(v)}$ describes the longest possible burst from higher priority VCANs. It is constraint by

$$\Theta_{hp(v)} \leq \sum_{\forall u \in hp(v)} \frac{b_u + r_u \Theta_{hp(v)}}{r_{phy}}, \tag{4}$$

which is equivalent to the transmission time of data admitted through tokens stored within the buckets and tokens that are added throughout $\Theta_{hp(v)}$. It poses an upper bound for the delay imposed by higher priority VCANs, because any other transmission requests from higher priority VCANs will not pass admission control at this point.

Solving (4) for $\Theta_{hp(v)}$ and inserting it in (3) gives an upper bound for the VCAN delay as

$$\Theta_v \leq C_{max,lp(v)} + \frac{\sum_{\forall u \in hp(v)} b_u}{r_{phy} - \sum_{\forall u \in hp(v)} r_u}. \tag{5}$$

After $t = t_0 + \Theta_v$, the depletion phase begins and the fill level will not reach another maximum. Therefore, sufficient bucket sizes for all VCANs can be calculated by using (2) with the upper bound for the VCAN delay derived from (5).

D. Fault Isolation and Error Handling

CAN is a fault-tolerant bus, i.e. it is capable of providing reliable communication in aggressive environments with high electromagnetic interference (EMI). In this Section, we analyze fault isolation and error handling in a virtualized CAN. We distinguish between faults introduced due to erroneous software behavior and faults that occur within the data layer or physical layer.

An unexpected software behavior can affect the operation of the CAN bus if it causes a deviation from the intended ID, cycle time or data length of a message. An alteration within most significant bits of the ID, which contain the VCAN tag, would allow senders to break out of their VCAN and affect other partitions. Therefore, these bits must not be modifiable during run-time. Any other changes to the ID cannot cause faults, which can cascade to other VCANs. An increase in data lengths or decrease in cycle times increases the required bandwidth of the sending node and its VCAN. An extreme case of such a fault is a 'babbling idiot' [20], which is a faulty node flooding the bus with messages. The proposed admission control module ensures that such behavior does not affect other VCANs. The faults presented so far are all capable of influencing the communication within its VCAN. However, the network virtualization ensures faults are constrained to the partition they originated from, resulting in a fault isolation among concurrent VCAN partitions.

Other faults do not originate from a specific VCAN partition, but rather occur within the data layer or physical layer. Bit-level errors caused e.g. by EMI occur at rates between 10^{-7} and 10^{-11} depending on the environment [21]. They are coped with by the CAN error handling and might require a retransmission of the frame. The necessity to retransmit reduces the utilizable bandwidth compared to the error-free case. Because no specific VCAN domain is accountable for such faults, these faults do not require to be isolated, but the effects should be handled to optimize the systems overall availability.

In mixed-criticality systems, highly critical functions are considered crucial to the availability of the entire system. Under the assumption that highly critical messages are mapped to high priority VCANs, we propose an error-handling, in which message retransmissions mainly affect low priority VCANs. This can be realized within the admission control by only reducing the tokens of a VCAN after successful transmissions. If a transmission fails in the highest priority VCAN, the retransmission can happen immediately. The retransmission of messages in low priority VCANs might get deferred by higher priority VCANs that gained additional tokens during the failed transmission attempt.

V. Real-time Analysis

In many applications of CAN, messages have to meet bounded deadlines. We will therefore show, how the maximum delay experienced by messages in virtual CANs can be calculated. The time span from issuing a message until its successful transmission on the bus will be called worst-case response time (WCRT) in the sequel. This analysis can be used to show the isolation properties necessary to enable mixed-criticality scenarios. We first introduce a conventional WCRT analysis for CAN and then extend the analysis for VCANs.

A. WCRT in conventional CAN

CAN is a non-preemptive, strict-priority bus for which a real-time analysis is presented in [22]. Because our analysis poses an extension to this work, it will briefly be introduced.

In a worst-case scenario for message m, all higher priority messages are issued at the same instant and the longest lower priority message has started transmission on the bus an infinitesimal time before.

The queuing delay w_m of message m describes the time from the issuing of the message until it wins the CAN arbitration. It is iteratively computed as

$$w_m^{n+1} = B_m + \sum_{\forall k \in hp(m)} \left\lceil \frac{w_m^n + J_k + \tau_{phy}}{T_k} \right\rceil C_k \tag{6}$$

and is composed of the blocking B_m and the transmission times of higher priority messages that are transmitted before message m. The maximum blocking is given as the transmission time of the longest lower priority message

$$B_m = max_{k \in lp(m)}(C_k). \tag{7}$$

The response time of each of these instances can be calculated by summation of queuing delay, jitter and message transmission time.

$$R_m = J_m + w_m + C_m. \tag{8}$$

The analysis can be extended to consider multiple instances of a message [22]. However, because the probability of this extended analysis being necessary is very low and only relevant for very high bus utilizations [23], we just consider the first instance of each message here.

B. WCRT in virtual CAN

In a virtual CAN, the transmission of messages is constrained in two additional ways: First, the average rate r_v, at which a VCAN is allowed to transmit is a fraction of the physical bandwidth r_{phy} and second, the communication flow of a VCAN can be delayed by other VCANs by up to Θ_v. In the following, we will present the worst-case scenario for a message transmission on a VCAN and derive a corresponding real-time analysis for message m in VCAN v.

For the worst-case scenario, traffic from other VCANs has to be assumed in such a way, that the interference maximizes the delay in the transmission of message m. This can be achieved, if higher priority VCANs have the maximum amount of tokens accumulated.

Similar to the scenario in conventional CAN, we assume that just when message m is about to get ready for transmission, the longest lower priority message within VCAN v starts transmitting on the bus. All higher priority messages within VCAN v are released synchronously with message m. At this point, the tokens had just reached the point, at which a transmission is possible. Even though the transmission time of the longest lower priority message on the bus is B_m, no transmission within VCAN v is possible for $B_m \cdot r_{phy}/r_v$, because only then, the tokens are recovered.

Additionally, the transmission of a message m can be blocked by other VCANs by up to Θ_v. Such a blocking scenario was already presented when deriving the bucket sizes b_v for each VCAN and is illustrated in Fig. 4. After such a scenario, VCAN v can transmit at an increased rate and will eventually equalize the blocking. Therefore, the maximum blocking occurs, if other VCANs start delaying VCAN v just before message m is eligible for transmission.

An upper bound for the possible delay can be calculated using (5). This delay cannot be exceeded, even if the burst scenario happens multiple times, because of the increased transmission rate afterwards. We therefore derive the increased blocking for message m in VCAN v as

$$\hat{B}_{m,v} = B_{m,v} \frac{r_{phy}}{r_v} + \Theta_v. \tag{9}$$

Using the increased blocking $\hat{B}_{m,v}$, the queuing delay can be calculated iteratively based on (6).

$$w_{v,m}^{\mu+1} = \hat{B}_{m,v} + \sum_{\forall k \in hp_v(m)} \left\lceil \frac{w_{v,m}^{\mu} + J_k + \tau_{phy}}{T_k} \right\rceil C_k \frac{r_{phy}}{r_v} \tag{10}$$

This equation takes the increased blocking as well as the reduced bandwidth into account. By introducing the factor r_{phy}/r_v, the transmission time is increased based on the reserved bandwidth of VCAN v. While the transmission of individual messages does not actually take longer, it has to be extended in the analysis, because during this duration, no transmission is possible in this scenario due to a lack of tokens. Using the queuing delay from (10) the WCRT R_m of message m can be calculated using the unmodified (8).

While the delay in the overall communication flow of messages in VCAN is bound by Θ_v, individual messages can experience bigger delays if additional higher priority messages get ready for transmission during the delay imposed by other VCANs. The effect of the increased blocking will be evaluated in the next Section.

VI. Analytic Evaluation

The relevance of the presented VCAN approach is tightly linked with its ability to provide latencies similar to those in physical CAN buses with the same bandwidth. Based on the analysis presented above, we computed latencies using a realistic traffic scenario for a setup with distinct physical buses and a setup, where these buses are consolidated as VCANs on a shared bus. We used the VCAN configuration described in Table I, which is equivalent to the one depicted in Fig. 2.

TABLE I
VCAN Configuration: Scenario 1

VCAN v	r_v	b_v	Θ_v
0	125 kbit/s	136 bit	0.27 ms
1	125 kbit/s	182 bit	0.63 ms
2	250 kbit/s	386 bit	1.27 ms

We randomly generated a message set, which utilizes 82% of the 125 kbit/s VCANs. The set is duplicated for the VCAN $v = 2$ with 250 kbit/s. Messages are cyclic and the cycle time distribution is based on actual in-car measurements presented in [24]. The cycle times T_m are depicted along with the results in Fig. 5.

The first row of Fig. 5 shows the message latencies in each VCAN compared to the case of dedicated physical buses. The additional latencies increase for decreasing VCAN priority. However, the qualitative development of the WCRT with respect to message priority indicates that VCANs show a temporal behavior very similar to physical CAN buses.

A more detailed evaluation is possible by analyzing the added latencies experienced in the VCAN setup, as depicted in the second row of Fig. 5. For VCAN $v = 0$, negative added latencies can be seen. The WCRT is 300 to 600 µs smaller for the VCAN. While the queuing delay w_m is

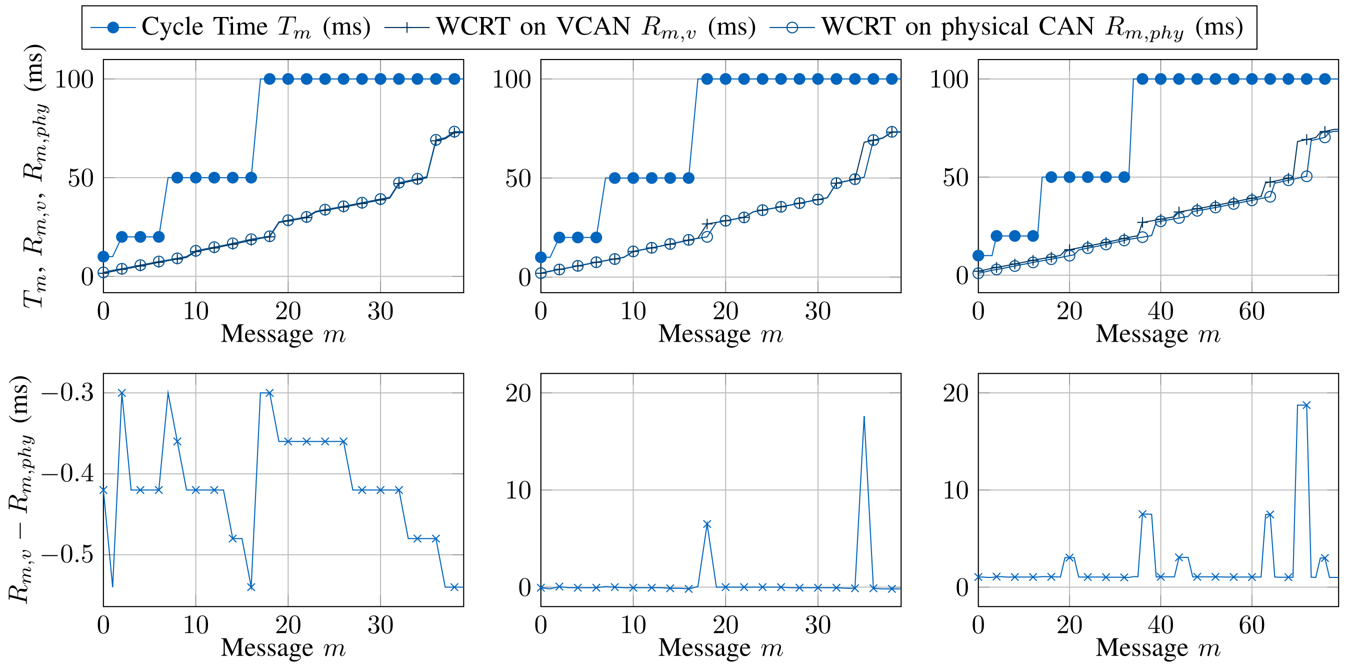

Fig. 5. Results from the analytical evaluation of a CAN with 3 concurrent VCANs. The upper row shows the WCRT for all messages and the lower row shows added latencies compared to an equivalent physical bus.

similar, each message m in VCAN v benefits from a faster transmission due to the higher physical bit rate. The effect is stronger for greater values of r_{phy}/r_v and for longer messages. Within VCAN $v = 0$, the variation in added latencies can be accounted to different message lengths.

The added latencies for VCAN 1 and 2 show peaks, which exceed 17 ms. Because messages in these VCANs can experience additional blocking when waiting for transmission, they can be overtaken by further instances of higher priority messages from the same VCAN, which get queued during this delay. This can happen for messages with a WCRT close to multiples of the cycle times of other messages. The highest peaks can be witnessed for messages with a WCRT around 50 ms, because here messages with 10 ms and 50 ms get queued for transmission.

TABLE II
VCAN CONFIGURATION:SCENARIO 2

VCAN v	r_v	b_v	Θ_v
0	100 kbit/s	135 bit	0.27 ms
1	100 kbit/s	169 bit	0.61 ms
2	100 kbit/s	237 bit	1.28 ms
3	100 kbit/s	406 bit	2.98 ms
4	100 kbit/s	1055 bit	9.47 ms

VII. EXPERIMENTS & RESULTS

In scenario 1, the added latencies do not cause the WCRTs of any message to exceed its cycle time, which is often equal to its deadline. A second scenario described by Table II shows the scalability limitations of the approach. The results depicted in Fig. 6 show an increase in added latencies for lower priority VCANs, which eventually leads to WCRTs for VCAN $v = 4$ that exceed their respective cycle times for a number of messages, indicating that deadline violations are possible. However, realistic application scenarios are unlikely to require hard real-time communication in all VCANs. Additionally, the probability for the occurrence of message response times close to their WCRT is decreasing with the VCAN priority, because of the increasing complexity of the assumptions made for the worst-case scenario in lower priority VCANs.

For soft-real time and best effort communication, no formal schedulability analysis is required. Rather, experimental measurements can be considered to determine the feasibility of the communication channel. Hence, we simulated an experiment in order to gain an estimation regarding average and maximum latencies experienced in VCANs. In the following, we will first introduce the experimental setup and afterwards, we present and discuss the obtained results.

A. Experimental Setup

To achieve comparable results, the traffic patterns used in the experiments are equivalent to the ones created for the analytic evaluation presented in Section VI. The VCAN configuration corresponds to the second scenario and features five VCANs with 100 kbit/s each.

The worst-case scenario presented makes pessimistic assumptions regarding the alignment of message release times and the bucket fill level of each VCAN. Due to e.g. unequal cycle times between messages and oscillator drifts between CAN nodes, these cases will not occur consistently and average latencies are much smaller than in these cases. Goal

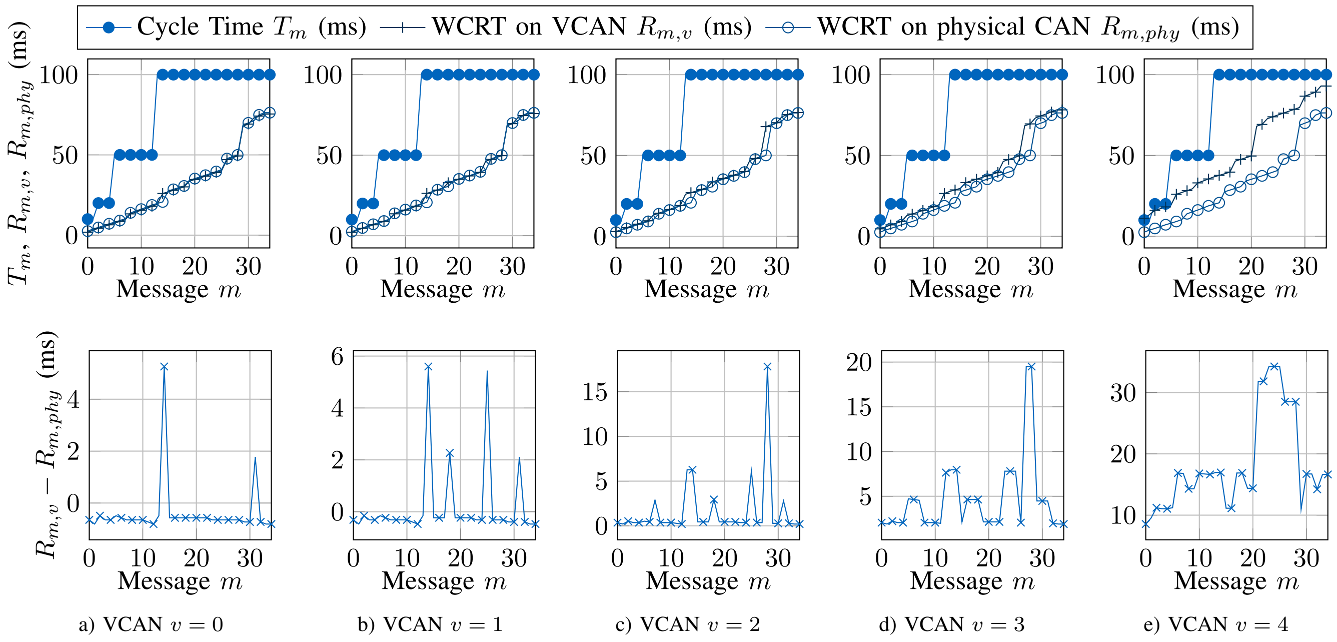

Fig. 6. Results from the analytical evaluation of a VCAN with 5 concurrent VCANs of equal bandwidth (100 kbit/s). The upper row shows the WCRT for all messages and the lower row shows added latencies compared to an equivalent physical bus.

of the experimental design is to gain measures, which reflect a realistic and average behavior.

We simulated 100,000 short experiments, each seeded with random values for the token bucket fill levels and for the first release of each message. The random numbers are uniformly distributed and include values between 0 bit and the bucket size b_v, and release times between 0 s and the cycle time T_m. After the first instance of a message, further instances are issued cyclic with respect to their cycle time. Each experiment covers 200 ms, which ensures that at least one instance of each message is transmitted. A longer simulation does not improve the results because possible message alignments have been covered. The total simulated time is around 5.5 h.

The main measures deducted are the average response time (time from release till successful transmission) of a message $\overline{R_{exp}}$ and the maximum response time measured $max(R_{exp})$. The simulation is carried out with a temporal resolution of 2 µs (equivalent to a CAN bit time), which guarantees bit accurate results.

B. Results & Discussion

The maximum and average latencies of all messages within every VCAN measured throughout the experiments are presented in Fig. 7. They are compared with the results obtained with a 100 kbit/s physical CAN bus using the same message set as the VCANs.

Throughout the experiments, the VCANs show a behavior, which is similar that of a distinct physical CAN bus, where maximum and average latencies increase with decreasing priority. However, the absolute values differ between physical and virtual CANs, as well as between VCANs of different priorities.

Fig. 7a shows that the maximum latencies of high priority messages are increasing with decreasing VCAN priority. As

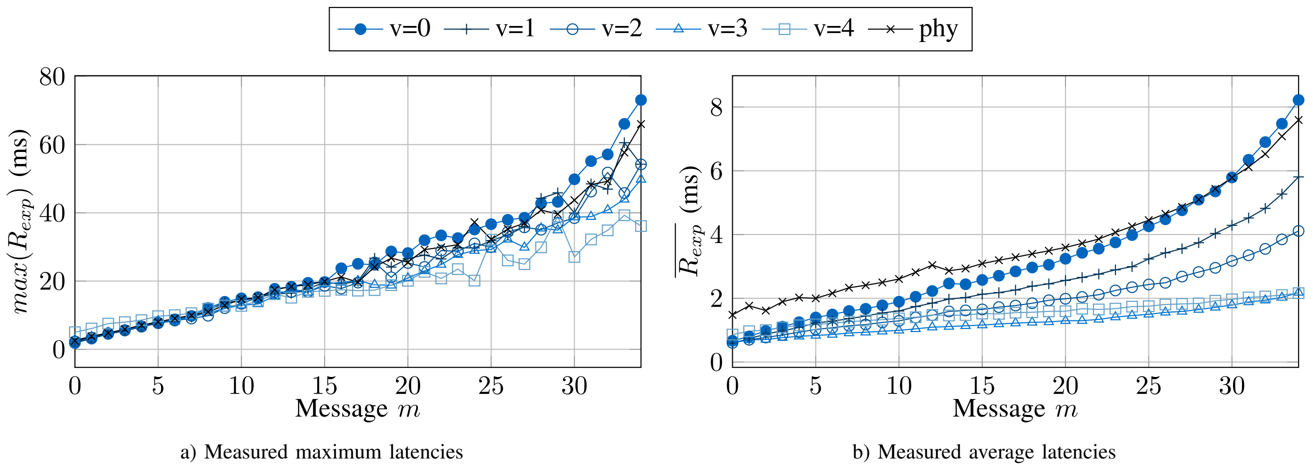

Fig. 7. Experimental results for 5 concurrent VCANs on a shared 500 kbit/s CAN bus compared to a 100 kbit/s physical CAN bus.

predicted in the analytic evaluation, the highest priority VCAN slightly outperforms the physical CAN, while high priority messages from lower priority VCANs suffer from an increasing blocking and therefore increased maximum latencies.

Despite the potential blocking from higher priority VCANs, low priority messages have lower maximum latencies when transmitted through low priority VCANs. Low priority VCANs have increased token bucket sizes to be able to cope with the blocking from higher priority VCANs. The increased bucket size enables more back-to-back transmissions from a single VCAN and is responsible for the decreased maximum latencies. The worst-case for lower priority VCANs is less probable, because it requires additional worst-case behavior from higher priority VCANs. The experiment suggests that the use of low priority VCANs is feasible in functional domains, where no formal verification is required.

The average measured latencies shown in Fig. 7b strengthen this observation. They develop similarly, where high priority messages in high priority VCANs have the best performance and low priority VCANs provide lower average latencies for low priority messages. Again, the increased token bucket sizes allow low priority VCANs to overcome the higher average blocking from higher priority VCANs.

The highest priority VCAN is the only one to eventually exceed the average latencies measured in the physical case. Because of its small token bucket, it cannot burst multiple messages on an idle bus and therefore benefits less from the resource sharing than lower priority VCANs. Compared to the physical case, low priority messages in VCAN $v = 0$ can be delayed by other VCANs, which allows them to be overtaken by high priority messages. While the same effect occurs in lower priority VCANs, the benefits of resource sharing outweigh these delays.

On average, the resource sharing enabled through network virtualization allows VCANs to outperform the physical CAN independent of the VCAN priority. The VCANs benefit from lower transmission times, because the actual physical bit rate is higher. Additionally, because other VCANs are not 100% utilized, the bus is idle more often, which opens up additional options for message transfers.

VIII. Implementation

We implemented an admission control module as extension to current CAN controllers. It constrains the transmission of messages to conform with the virtualization concepts presented in this paper. In the following, we present an architectural description and implementation results.

Fig. 8 shows the architectural view of the module. Its inputs are exclusively signals that are derived from the CAN bus. Therefore, these signals are consistent in every CAN node on the same physical bus. The only output of the module *tx_allowed* indicates, whether a potential transmission conforms with the admission control rules.

The submodules *time_counter* and *vcan_lookup* provide essential input to the *bucket_mgmt* module. The signals *id* and *ide* (ID extended) represent the message ID of the frame currently transferring, from which the VCAN ID is extracted by *vcan_lookup*. In this exemplary implementation, the *vcan* signal consists of three bits and therefore supports 8 VCANs.

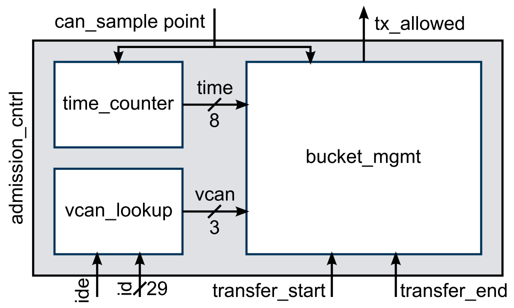

Fig. 8. Block diagram of the admission control module

Each rising edge in *can_sample_point* is used to increment a counter within the *time_counter* submodule to create a time value with the resolution of a CAN bus bit time τ_{phy}. In order to reliably measure the transmission time of a message, it must not be possible for the counter to overrun more than once during a single transfer. With a signal width of 8 bit, it can represent values within the range of $\{0...255\}\tau_{phy}$. This is sufficient to guarantee correct operation, because the maximum value exceeds the worst-case transmission time of a maximum length message (160 τ_{phy} for extended IDs).

The main part of the admission control is done by the *bucket_mgmt* module. It maintains the token bucket and constrains message transmission. After each n-th sample point, a token is added to the bucket, where n is given as $n = r_v/r_{phy}$. Therefore, only integer values for r_v/r_{phy} are possible with this implementation. At the start of a message transfer, indicated by a rising edge in *transfer_start*, a time stamp is taken. By comparison with the *time* signal after a successful transfer indicated by *transfer_end*, the transmission time of the last transfer can be computed. If the transfer happened within its own VCAN, tokens equivalent to the actual transmission time will be removed from the bucket. Because bucket size might temporarily be exceeded, the tokens will be adjusted at the beginning of each transmission. The signal *tx_allowed* is logically high if the fill level of the bucket exceeds the minimum amount necessary for transmission.

The module can easily be integrated with existing CAN controllers. They usually feature a user-programmable bit to indicate a Tx request. A logical AND connection with *tx_allowed* allows to restrict transmissions in conformance with the admission control mechanism.

We integrated the admission control module together with an OpenCores CAN controller, which is functionally equivalent to the Philips/NXP SJA1000. The design was prototypically implemented for Spartan-6 FPGAs. In order to achieve

TABLE III
Hardware Resource Requirements

module	Area	NAND2 equivalents
Standard CAN Controller	81355.04 μm^2	14711.58
VCAN Controller	82668.54 μm^2	14949.10

comparable figures regarding the hardware requirements of the extensions, we synthesized the standard and the extended CAN controller at 24 MHz using the Synopsis 90 nm generic library. The results presented in Table III show the area demand in both implementations measured in area and normalized to the size of a NAND2 gate in this technology (5.53 μm^2). The synthesis results show that the extensions for network virtualization support require only 1.61% additional area.

IX. Conclusion

To enable future integrated automotive IT architectures, we developed a network virtualization approach for Controller Area Network (CAN). In this paper, we presented a token bucket based admission control scheme, which guarantees isolated performance for concurrent virtual CANs (VCANs) on a shared physical bus. The concept focuses on providing real-time capable communication in mixed-criticality scenarios and provides fault isolation among the partitions.

We showed that the worst-case delay of VCAN communication flows is bounded. Based on an extended worst-case response time analysis for VCANs, we evaluated the performance with respect to message latencies in different VCAN setups. Because latencies for low priority VCANs increase with the total number of VCANs, message sets might be schedulable under real-time constraints. Using a realistic automotive workload, we showed that deadline violations can occur in the lowest priority VCAN for when using 5 VCANs. However, automotive application scenarios are not expected to require hard real-time guarantees in all partitions.

Additionally, we evaluated the average and maximum latencies in a simulated experiment. The experiments show that the VCANs outperform its physical equivalent in average latencies and show very similar maximum latencies. The results show that scalability limitations can be overcome if not all VCANs require hard real-time communication, which is common in mixed criticality scenarios.

We presented a prototypical implementation of the extensions required for the network virtualization. The synthesis in 90 nm technology showed that the network virtualization extensions increase the area demand of a standard CAN controller by 1.61%.

References

[1] H.-U. Michel, "Taming multicores for safe transportation: Aramis in the automotive domain," in *Workshop on the Integration of mixed-criticality subsystems on multi-core processors, presented at HiPEAC 2013*, 2013.

[2] D. Reinhardt and M. Kucera, "Domain controlled architecture: A new approach for large scale software integrated automotive systems," in *Pervasive and Embedded Computing and Communication Systems*, 2013, pp. 221–226.

[3] M. Di Natale and A. L. Sangiovanni-Vincentelli, "Moving from federated to integrated architectures in automotive: The role of standards, methods and tools," *Proceedings of the IEEE*, vol. 98, no. 4, pp. 603–620, 2010.

[4] D. Reinhardt, D. Kaule, and M. Kucera, "Achieving a scalable e/e-architecture using autosar and virtualization," *SAE International Journal of Passenger Cars-Electronic and Electrical Systems*, vol. 6, no. 2, pp. 489–497, 2013.

[5] S. Trujillo, A. Crespo, and A. Alonso, "Multipartes: Multicore virtualization for mixed-criticality systems," in *Digital System Design (DSD), 2013 Euromicro Conference on*. IEEE, 2013, pp. 260–265.

[6] R. Pellizzoni, P. Meredith, M.-Y. Nam, M. Sun, M. Caccamo, and L. Sha, "Handling mixed-criticality in soc-based real-time embedded systems," in *Proceedings of the seventh ACM international conference on Embedded software*. ACM, 2009, pp. 235–244.

[7] S. Tobuschat, P. Axer, J. Diemer, and R. Ernst, "Idamc: A noc for mixed-criticality systems," in *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2013 19th IEEE International Conference on*, 2013.

[8] S. Baruah, H. Li, and L. Stougie, "Towards the design of certifiable mixed-criticality systems," in *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2010 16th IEEE*. IEEE, 2010, pp. 13–22.

[9] A. Burns and R. Davis, "Mixed criticality on controller area network," in *Real-Time Systems (ECRTS), 2013 25th Euromicro Conference on*. IEEE, 2013, pp. 125–134.

[10] N. Chowdhury and R. Boutaba, "A survey of network virtualization," *Computer Networks*, vol. 54, no. 5, pp. 862–876, 2010.

[11] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang, "Secondnet: A data center network virtualization architecture with bandwidth guarantees," in *Proceedings of the 6th International COnference*, ser. Co-NEXT '10. New York, NY, USA: ACM, 2010, pp. 15:1–15:12.

[12] B. Raghavan, K. Vishwanath, S. Ramabhadran, K. Yocum, and A. C. Snoeren, "Cloud control with distributed rate limiting," in *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4. ACM, 2007, pp. 337–348.

[13] L. Zhao, M. Li, Y. Zaki, A. Timm-Giel, and C. Görg, "Lte virtualization: from theoretical gain to practical solution," in *Proceedings of the 23rd International Teletraffic Congress*. ITCP, 2011, pp. 71–78.

[14] J. Flich, S. Rodrigo, J. Duato, T. Sodring, A. Solheim, T. Skeie, and O. Lysne, "On the potential of noc virtualization for multicore chips," in *International Conference on Complex, Intelligent and Software Intensive Systems (CISIS)*. IEEE, 2008, pp. 801–807.

[15] J. Heisswolf, A. Zaib, A. Weichslgartner, R. König, T. Wild, J. Teich, A. Herkersdorf, and J. Becker, "Virtual networks – distributed communication resource management," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 6, no. 2, pp. 8:1–8:14, Aug. 2013.

[16] J. Kim, S. Lee, and H. Jin, "Fieldbus virtualization for integrated modular avionics," in *Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conference on*. IEEE, 2011, pp. 1–4.

[17] C. Herber, A. Richter, H. Rauchfuss, and A. Herkersdorf, "Self-virtualized can controller for multi-core processors in real-time applications," in *International Conference on Architecture of Computing Systems (ARCS)*, 2013, pp. 244–255.

[18] G. Leen and D. Heffernan, "Ttcan: a new time-triggered controller area network," *Microprocessors and Microsystems*, vol. 26, no. 2, pp. 77–94, 2002.

[19] G. Niestegge, "The leaky bucketpolicing method in the atm (asynchronous transfer mode) network," *International Journal of Digital & Analog Communication Systems*, vol. 3, no. 2, pp. 187–197, 1990.

[20] I. Broster and A. Burns, "An analysable bus-guardian for event-triggered communication," in *Real-Time Systems Symposium, 2003. RTSS 2003. 24th IEEE*. IEEE, 2003, pp. 410–419.

[21] J. Ferreira, A. Oliveira, P. Fonseca, and J. Fonseca, "An experiment to assess bit error rate in can," in *Proceedings of 3rd International Workshop of Real-Time Networks (RTN2004)*. Citeseer, 2004, pp. 15–18.

[22] 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.

[23] M. Di Natale and H. Zeng, "Practical issues with the timing analysis of the controller area network," in *Emerging Technologies & Factory Automation (ETFA), 2013 IEEE 18th Conference on*. IEEE, 2013, pp. 1–8.

[24] B. Müller-Rathgeber, M. Eichhorn, and H.-U. Michel, "A unified car-it communication-architecture: Design guidelines and prototypical implementation," in *Intelligent Vehicles Symposium, 2008 IEEE*. IEEE, 2008, pp. 709–714.