Security-aware Obfuscated Priority Assignment for Automotive CAN Platforms

MARTIN LUKASIEWYCZ, TUM CREATE Limited
PHILIPP MUNDHENK, TUM CREATE Limited
SEBASTIAN STEINHORST, TUM CREATE Limited

Security in automotive in-vehicle networks is an increasing problem with the growing connectedness of road vehicles. This paper proposes a security-aware priority assignment for automotive Controller Area Network (CAN) platforms with the aim of mitigating scaling effects of attacks on vehicle fleets. CAN is the dominating field bus in the automotive domain due to its simplicity, low cost, and robustness. While messages might be encrypted to enhance the security of CAN systems, their priorities are usually identical for automotive platforms, comprising generally a large number of vehicle models. As a result, the identifier uniquely defines which message is sent, allowing attacks to scale across a fleet of vehicles with the same platform. As a remedy, we propose a methodology that is capable of determining obfuscated message identifiers for each individual vehicle. Since identifiers directly represent message priorities, the approach has to take the resulting response time variations into account while satisfying application deadlines for each vehicle schedule separately. Our approach relies on Quadratically Constrained Quadratic Program (QCQP) solving in two stages, specifying first a set of feasible fixed priorities and subsequently bounded priorities for each message. With the obtained bounds, obfuscated identifiers are determined, using a very fast randomized sampling. The experimental results, consisting of a large set of synthetic test cases and a realistic case study, give evidence of the efficiency of the proposed approach in terms of scalability. The results also show that the diversity of obtained identifiers is effectively optimized with our approach, resulting in a very good obfuscation of CAN messages in in-vehicle communication.

Categories and Subject Descriptors: C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: CAN, priority assignment, automotive, security

ACM Reference Format:
Martin Lukasiewycz, Philipp Mundhenk, and Sebastian Steinhorst, 2015. Security-aware Obfuscated Priority Assignment for Automotive CAN Platforms. *ACM Trans. Des. Autom. Electron. Syst.* 0, 0, Article 00 (2015), 27 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

1.1. Automotive CAN Security

Automotive networks that comprise up to 100 Electronic Control Units (ECUs) and multiple bus systems are a result of an incremental design process over the past 25 years [Di Natale and Sangiovanni-Vincentelli 2010]. This process is equally driven both by the necessity to innovate and very tight budget constraints. As a matter of fact, these

This work was financially supported by the Singapore National Research Foundation under its Campus for Research Excellence And Technological Enterprise (CREATE) programme.
Author's addresses: M. Lukasiewycz and P. Mundhenk and S. Steinhorst, TUM CREATE Limited, Singapore.

DOI: http://dx.doi.org/10.1145/0000000.0000000

Fig. 1. Illustration of a CAN frame. The identifier is used for arbitration and, therefore, cannot be encrypted while the data might be encrypted. As a result, the identifier exposes information about the type of data to attackers due to the fixed coupling of identifiers and data in the automotive domain. The used acronyms are: SOF - Start of Frame, RTR - Remote Transmission Request, CRC - Cyclic Redundancy Check, ACK - Acknowledgment Slot, EOF - End of Frame.

networks used to be isolated in nature without any permanent communication interface to external devices. Due to changing customer requirements, modern vehicles have to be cloud-connected and integrate functions such as smart routing, advanced infotainment, etc. This, however, leads to potential threats as networks that initially have not been designed with security in mind are exposed to access from external devices.

One approach to decrease the security risks is the integration of firewalls in those devices that have an external connection. However, once these devices are compromised or bypassed, the bus systems and their data are exposed. Considering these security risks in the automotive domain, the large number of vehicles that might be affected by an attack has to be taken into account. As vehicle models are often using a common hardware/software platform in order to reduce development efforts [Wilhelm 1997], an attack can be executed with minimal or no changes on a large number of vehicles. This scaling multiplies the effects of any remote attack and makes large scale attacks on automobiles with comparatively small efforts feasible.

In todays vehicles, multiple different networks are used. While FlexRay [FlexRay Consortium 2005] and Ethernet [Bello 2011] can provide high bandwidth and determinism, CAN [Bosch 1991] is still the prevalent bus system in the automotive domain. CAN is a shared bus standard that relies on a fixed-priority non-preemptive communication, providing a very cheap and reliable solution for distributed functions. While data transmission over CAN is susceptible to congestion, it may be used for real-time functions like Anti-lock Braking System (ABS) or Electronic Stability Program (ESP) that require strict end-to-end latencies. By considering all streams on the bus at design time, it is possible to determine the exact worst-case transmission time of any message. A CAN frame is illustrated in Figure 1. CAN has a data-rate of up to $1\,\mathrm{Mbit/s}$ and a maximal payload of $8\,\mathrm{bytes}$ per message. The identifier field is $11\,\mathrm{bits}$ ($29\,\mathrm{bits}$ for extended CAN) and used for arbitration such that messages with lower values are always prioritized in case of bus contention. The arbitration makes use of the binary representation of priorities and the fact that the transmission of a logical *0* is dominant on the bus. It should be noted that identifiers have to be unique to prevent conflicts. In this paper we will use *identifier* and *priority* of a CAN message interchangeably.

Recent developments towards CAN with Flexible Data-Rate (FD) [Hartwich 2012] increase the payload of a single frame up to $64\,\mathrm{bytes}$, enabling for instance security features like encryption and key exchanges. While encryption can protect data particularly with a higher data-rate using CAN FD, the message headers have to remain unencrypted as the arbitration method requires a binary encoding. Therefore, attackers might obtain or manipulate vehicle data that is transmitted over the shared bus once a single device is compromised and when the mapping from message identifiers to signals is fixed and known. Retrieving information about this relation between identifiers and data can be carried out by an attacker for a specific platform via the On-Board Diagnostics (OBD) port and reverse engineering, using unlimited resources in terms of time and computational power. Since CAN message priorities are static and identical

for a certain vehicle platform, the obtained information can be directly applied to an entire fleet of vehicles to implement a potential exploit.

As a remedy, we propose an approach that allows to determine specific message identifiers per vehicle for CAN systems while still satisfying all real-time end-to-end latencies. Thus, the relation between identifier and data is different for each vehicle and attackers have to face an additional obstacle when implementing exploits that are supposed to affect an entire fleet of vehicles.

1.2. Contributions of the Paper

This paper proposes a novel approach to improve the security of CAN buses in the automotive domain. Orthogonally to encryption of messages and firewalls, we propose an obfuscation of message identifiers of CAN buses for vehicle platforms. By obfuscating identifiers, the proposed approach allows to break the scaling effects of attacks on vehicle fleets that share the same platform. To the best of our knowledge, this is the first priority optimization approach that considers scaling effects on vehicle fleets.

Since the obfuscation is determined at design time, additional security comes at no cost at run time using the existing platform. At the same time, it has to be taken into account that embedded devices in vehicles are often highly constrained in terms of computational power and memory. Thus, the proposed economic solution is particularly relevant for the automotive domain where large production numbers mandate an efficient implementation of security mechanisms. As a result, the proposed approach forms an obstacle for compromised devices inside the vehicle to retrieve information about or manipulate CAN data, improving the effective confidentiality and integrity.

In Section 2, we give an overview of the proposed methodology and explain the main design flow as well as the advantages and integration challenges for the approach. We also present and discuss the graph-based system model and the basic response time equations that are used throughout the paper. The methodology is based on three stages:

— In Section 3, a QCQP approach is proposed that is capable of determining fixed priorities for an automotive network consisting of CAN buses and ECUs with a fixed-priority real-time operating system. We improve the approach from [Lukasiewycz et al. 2013] by avoiding an intermediate graph-representation and explicitly allowing global cycles while determining the priorities per resource. The resulting comprehensive formulation efficiently optimizes the average slack of tasks and serves as basis for our bounded priority determination.
— The approach that is capable of determining priority bounds is proposed in Section 4. Given a set of fixed priorities from the first stage, a QCQP is presented that is capable of optimizing the bound range of message streams in terms of their priorities. That means, the priority of any message can be chosen from a determined range while all potential resulting schedules still satisfy existing deadline constraints.
— Given the priority bounds for messages, Section 5 presents a Monte Carlo method to determine the specific priority assignments for each vehicle. The method in this final stage is based on sampling and sorting such that for all test cases obfuscated priorities are determined in less than $1\,\mathrm{ms}$. This short runtime makes it suitable to obtain obfuscated CAN identifiers for an entire fleet of vehicles.

In Section 6, experimental results for a set of 500 test cases and one realistic case study are presented and discussed. The results show that the proposed approach is very efficient and capable of handling large systems with many ECUs and multiple CAN buses. It is also illustrated how the average slack optimization increases the bound range of priorities and how this in turn increases the priority diversity.

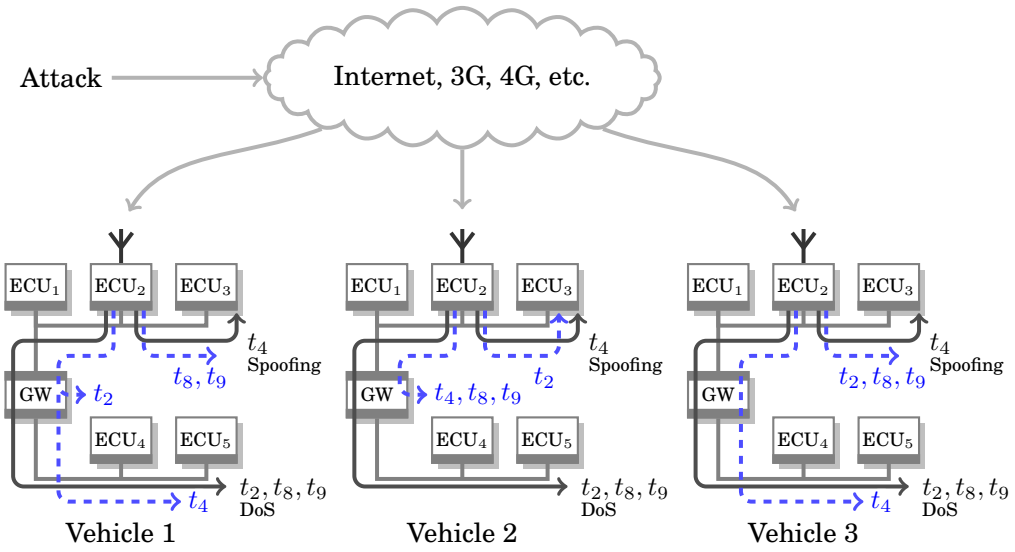

Fig. 2. Illustration of an attack on a vehicle fleet via an exploited device ECU_2 that has access to a CAN bus and is able to send messages. If there exists a fixed priority assignment (⟶) , attacks that comprise *spoofing* or *Denial-of-Service* (DoS) might scale across the entire fleet. With priority obfuscation (⇢) , the attack does not scale as the priorities or identifiers, respectively, have a different designated purpose in each vehicle.

In Section 7, related work in the domain of CAN scheduling and automotive security is discussed. Finally, the paper is concluded in Section 8.

2. METHODOLOGY

In the following, the proposed approach is outlined. First, we discuss attack scenarios and how these can scale across an entire fleet. Using the proposed obfuscation of CAN priorities, it is explained how the scaling effects can be mitigated. Subsequently, we outline our design approach and discuss integration and maintenance challenges that arise in case of individual CAN priorities for each vehicle. Finally, the system model and used response time equations are introduced.

2.1. Attack Scenarios and Obfuscation

With the growing connectedness of road vehicles, the susceptibility to cyber-attacks is rapidly increasing. A particular danger lies in scaling effects where one attack can affect an entire fleet of vehicles that rely on the same hardware/software platform. This scenario is illustrated in Figure 2: With fixed priorities that are the same for each vehicle in the fleet, a compromised device like ECU_2 that has access to a CAN bus can carry out an attack by sending particular messages. In one scenario, the compromised device can perform spoofing by sending messages with specific identifiers. This is illustrated in Figure 2 by ECU_2, sending the unencrypted message t_4 to ECU_3 and manipulating its state. Note that in this example, message t_4 is supposed to originate from a different device than ECU_2. An example of the consequences of this spoofing attack might be the overwriting of the vehicle speed on the instrument cluster which might result in threats to the safety of all traffic participants. In another scenario, a Denial-of-Service (DoS) attack can be carried out by sending a high amount of messages with high priorities, making the bus unavailable to other devices. Such an attack is illustrated in Figure 2 where ECU_2 sends the messages t_2, t_8, and t_9 to another bus via

the gateway, affecting the functionality of the safety-critical devices ECU_4 and ECU_5. In order to succeed, the attacker has to know which messages are forwarded within the gateway and in order to circumvent traffic shaping, a combination of different messages has to be sent. This employed DoS attack might directly influence safety-critical functions like ABS and ESP, even if the used messages are encrypted. These two examples illustrate the potential threats with respect to integrity and availability which due to the same priority assignment in vehicles scale across the entire fleet. In the following it is discussed how an obfuscation of message priorities can prevent these scaling effects.

The discussed scaling attacks can be mitigated by traditional security countermeasures. The encryption of messages can prevent spoofing while a firewall in the gateway can be used to avoid the DoS attack. In this context, the proposed priority obfuscation method is orthogonal to established methods such as encryption and firewalls. At the same time, the operational costs of the proposed approach are lower than encryption or firewalls which require a significant amount of additional hardware and software. On the other hand, it should be mentioned that the proposed method requires additional efforts in configuration and maintenance which is discussed subsequently in this section.

In case of obfuscated priorities, the described attacks might be mitigated since the identifiers of messages change from vehicle to vehicle as illustrated in Figure 2. The spoofing attack loses its effectiveness as the compromised device first would have to learn the correct identifier. In the example in Figure 2 merely the second vehicle (Vehicle 2) receives a message from the compromised ECU_2 while the other two vehicles are not affected at all. As a result, this attack does not scale. The DoS attack is as well not effective any more since the knowledge on which message is forwarded to the other bus is not available. For all three vehicles in Figure 2, the messages are either not forwarded to the second bus or they are forwarded to another bus such that only t_4 is existent on the second bus in Vehicle 1 and 3. As a result, the traffic of only t_4 might not be enough to carry out a successful DoS attack on the second CAN bus. Note that a DoS attack on the adjacent bus of ECU_2 cannot be prevented with the proposed obfuscation methodology. However, in practice devices that have external connections like ECU_2 are integrated in different domains than safety-critical devices like ECU_4 and ECU_5 and, thus, are separated by a gateway.

The attack might be improved by letting the compromised device identify the bus traffic, using system identification approaches such as [Di Natale and Zeng 2010]. However, sophisticated approaches to identify the traffic require computational power and memory which can be highly constrained in ECUs. Another obstacle is the time window in a large scale attack which might be very short as all vehicles would be affected and with dedicated or virtual honeypots [Spitzner 2003], an Original Equipment Manufacturer (OEM) could detect these attacks quickly. While dormant viruses are easy to implement on general purpose computers with a lot of (unused) resources, dedicated embedded platforms are easier to monitor for the OEMs in order to detect large scale attacks as soon as possible.

It should be further mentioned that [Di Natale and Zeng 2010] might be applied to identify the periods of messages and clock drifts of devices which could be used for grouping of messages. This, however, would not be sufficient to map from obfuscated priorities to actual priorities as many messages have the same period. Particularly, when message splitting is applied as proposed in Section 5, a detailed system identification with constrained resources can as a consequence be very time intensive. Thus, priority obfuscation is an effective obstacle against attacks in automotive systems at low cost which might be applied complementary to existing encryption and firewalls. The proposed approach can be compared to address space randomization [Shacham

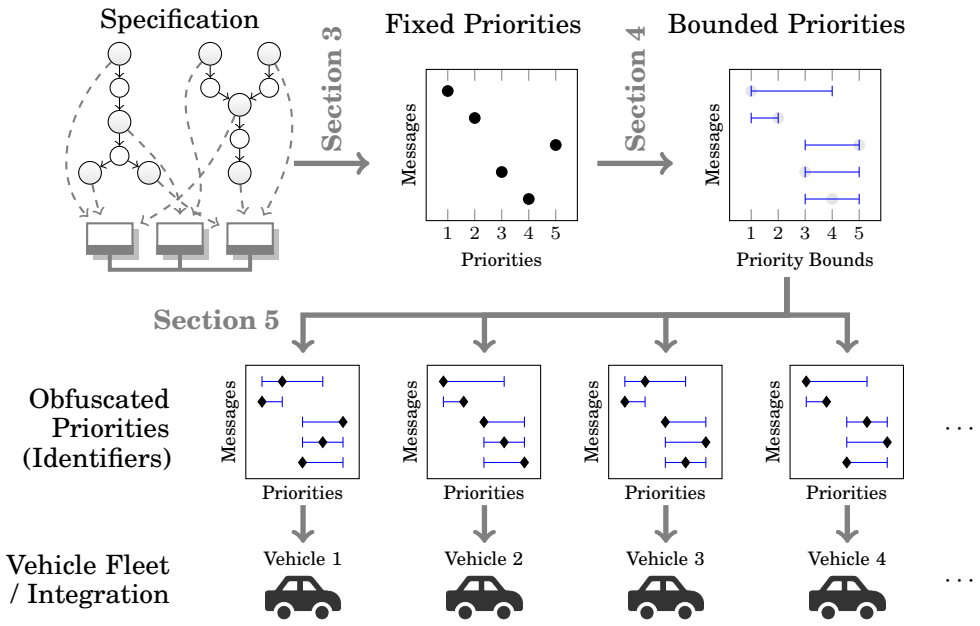

Fig. 3. Illustration of the proposed approach. Fixed priorities are determined from the system specification, followed by a determination of bounded priorities. The bounds are finally used to obtain obfuscated fixed priorities for the vehicles of a fleet.

et al. 2004] which is widely used to mitigate buffer overflow attacks by randomizing the memory space. While the randomized memory space can be defeated as described in the work mentioned above (under investment of significant computational power and time), it has proved to be an effective obstacle against attacks.

2.2. Design Approach

The proposed approach to obtain obfuscated priority assignments efficiently is illustrated in Figure 3. In the first stage, the system specification is used to determine fixed priorities for tasks and messages such that all end-to-end latencies are satisfied. Note that priorities are representing corresponding CAN identifiers. The priorities are determined such that the average slack of all tasks is optimized. This optimization is carried out with a QCQP solver, relying on an efficient constraint formulation of the underlying problem. In the next stage, bounded priorities for messages are determined while the priorities of tasks are fixed. Here, each message stream is defined by a lower bound and upper bound priority which are chosen such that all deadlines are still satisfied. By modifying the constraint formulation from the fixed priority optimization, a QCQP solver is used to maximize the average bound span which is defined as the difference between the lower and upper bound priority. Note that both QCQPs are solved directly without any relaxation or approximation. Using the bounds on message priorities, in the final stage it is possible to obtain obfuscated priorities for each vehicle. For this purpose, an efficient sampling and sorting approach is applied.

The three stages of our methodology are detailed in Section 3, Section 4, and Section 5. It might be possible to combine the first and second stage into a single QCQP problem. However, this would require a significantly more complex problem formulation and, for

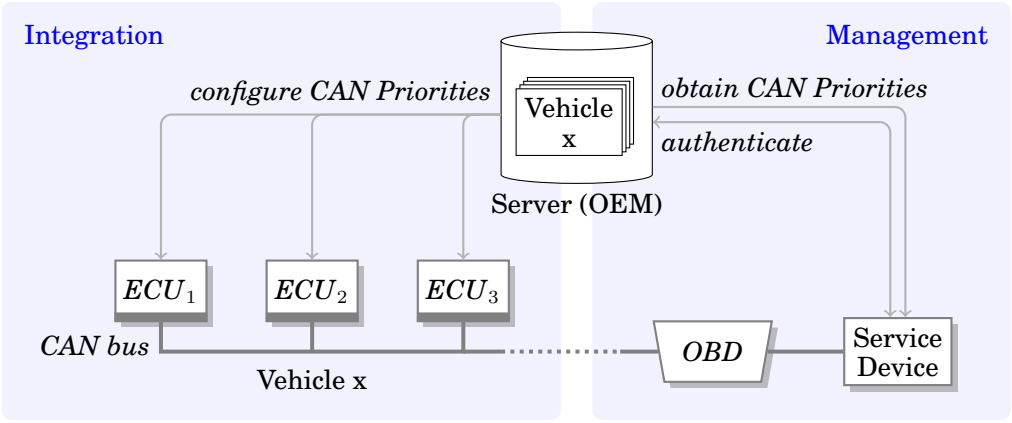

Fig. 4. Illustration of the integration and management challenges for the implementation of the proposed obfuscation of CAN priorities. Note that the integration is performed once during the configuration of the vehicle while the management might be carried out anytime during operation.

the sake of reducing the complexity and improving scalability, the stages are optimized separately.

A critical factor for the proposed approach is the priority determination complexity. Determining fixed priorities for a fleet of vehicles with a tool-based approach may take many hours, which is generally acceptable. However, when the same approach is used to obtain a specific priority assignment for each vehicle, it has to be taken into account that the runtime scales linearly with the number of vehicles, leading to possibly unacceptable runtimes. Therefore, the proposed determination of obfuscated priorities has to be very efficient. The first two stages of our approach from the specification to the bounded priorities may still have a runtime of a few hours as they are performed only once. In the final stage that determines an obfuscated priority assignment for each vehicle, a Monte Carlo sampling and sorting is used that turns out to be very fast and robust. Thus, the requirements regarding a fast priority determination are satisfied.

2.3. Integration and Management

For the practical use of the proposed obfuscation of priorities, additional efforts in integration and management become necessary that are outlined in the following. We use Figure 4 to illustrate the necessary infrastructure to implement the proposed methodology from an operational point of view.

The integration of obfuscated priorities requires the separate configuration of each vehicle as well as a server infrastructure where the information about each vehicle is stored securely as illustrated in Figure 4. Note that the integration step is performed uniquely before the delivery of the respective vehicle. For each ECU, the message filter for the CAN controller as well as the internal mapping of messages to signals have to be configurable. Since the integration involves ECUs that are developed by various suppliers, a unified interface has to be in place. This interface might rely on the Universal measurement and Calibration Protocol (XCP) [Caliebe et al. 2011] with additional authentication functions to ensure that the priorities can only be changed by the OEM. Similar approaches will have to be applied for encryption of messages in in-vehicle networks where each ECU will require a unique set of encryption keys. Since this configuration requires only a small amount of data, it is possible to perform it efficiently within a few seconds for one vehicle and wireless approaches might further enhance this process.

With obfuscated priorities, testing might be carried out with the known worst-case end-to-end function response times by delaying messages. This delaying of messages might be achieved either by a modification of the transceiver or additional messages might be sent over the CAN bus with a separate device. Another approach might be to determine a superset of configurations that need to be tested in order to obtain complete coverage.

In terms of management of individual priorities per vehicle which involves diagnosis, a service device has to obtain the CAN priorities for a certain vehicle as illustrated in Figure 4. For this purpose, the service device has to perform an authentication. Even if a service device is compromised, the amount of vehicle data that is provided per device can be limited by a reasonable quota to reduce the impact of a potential security threat significantly.

It should be noted that the mentioned data management and integration challenges are not in the scope of this paper and suitable approaches become as well necessary for future architectures when encryption of messages is widely used. Exemplary, a suitable approach for encryption from the domain of vehicle-to-vehicle communication is illustrated in [Harding et al. 2014, p. 167].

2.4. Specification

In this paper, we use a generic task graph as an application model and its mapping to an existing architecture. A task graph is defined as $G_T(T, E_T)$ where T is a set of tasks that also include messages. Note that in the following the terms task and message are used interchangeably. The set of edges in the task graph E_T defines data-dependencies such that for each $(t, \tilde{t}) \in E_T$, the task $\tilde{t}$ is released (ready for execution) when the task t finished its execution. Thus, an event-driven activation is assumed while the approach might be as well extended to periodic activations as discussed shortly in the constraint definition of the end-to-end latencies in the next section.

Each task in T is executed on exactly one resource in the set $A = A_{\text{ECU}} \cup A_{\text{CAN}} \cup A_{\text{GW}}$ that comprises ECUs, CAN buses, or gateways. For the case that messages are sent via multiple buses and gateways, each message stream is split into one task per resource in the system model. Note that the model might be extended with an arbitrary number of different components while we restrict it to ECUs, CAN buses, and gateways for the sake of simplicity. All tasks that are executed on a resource $a \in A$ are given as the set $T_a \subseteq T$.

It is assumed that the worst-case execution time e_t (equiv. to transmission time for messages) is known at design time. Moreover, each task has a period defined by h_t and it has a release jitter j_t. The release jitter is either defined by the source like a sensor or it is determined by the triggering of preceding tasks. An example of a task graph and a mapping to a resource architecture is given in Figure 5.

2.5. Scheduling and Response Times

To determine the worst-case response times of processes or messages, respectively, we use the recurrence equations from [Joseph and Pandya 1986] and [Davis et al. 2007], respectively. They are of the form (1) and (2). In this context, the function $p : T \to \mathbb{N}$ obtains the priority of a message or task. Here, a lower value indicates a higher priority.

Preemptive Scheduling. Following the approach in [Joseph and Pandya 1986], the maximum response time r_t of a task running on a resource with a preemptive scheduler, e.g., an ECU $a \in A_{\text{ECU}}$ with an operating system using fixed priorities[1], might be

[1]For the sake of simplicity, this paper does not differentiate between *runnables* and *tasks* as proposed in AUTomotive Open System ARchitecture (AUTOSAR) [AUTOSAR GbR 2014].

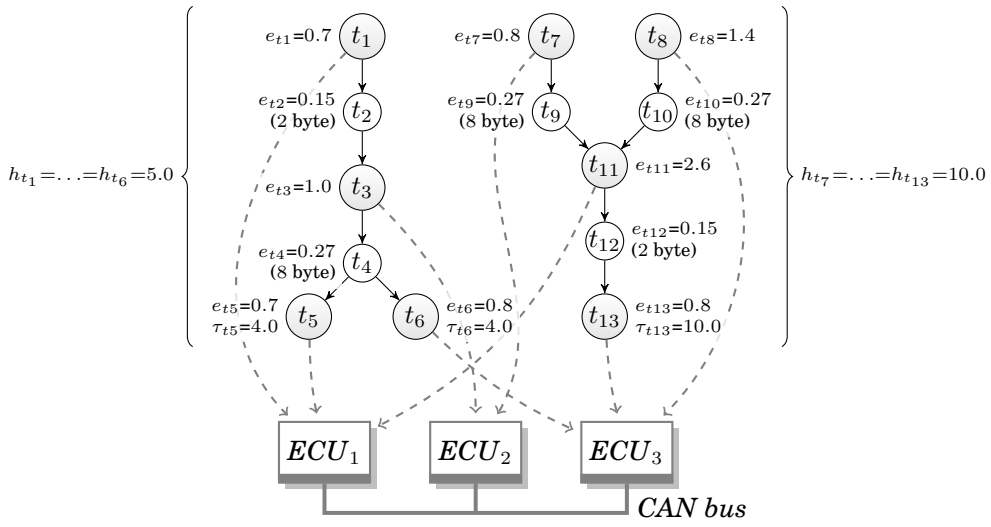

Fig. 5. Example of a task graph $G_T(T, E_T)$, consisting of two functions and a mapping to the resource set A that comprises three ECUs and one CAN bus. Note that the tasks t_4, t_9, t_{10} are 8 byte messages and t_2, t_{12} are 2 byte messages, respectively, mapped on the CAN bus with a bandwidth of $500kbit/s$. Annotated are the worst-case execution times e_t, deadlines τ_t, and periods h_t.

determined. Here, all possible preemptions of tasks with a higher priority have to be taken into account:

$$r_t = e_t + \sum_{\substack{\tilde{t} \in T_a \\ p(\tilde{t}) < p(t)}} \left\lceil \frac{r_t + j_{\tilde{t}}}{h_{\tilde{t}}} \right\rceil \cdot e_{\tilde{t}} \tag{1}$$

The response time is defined as the sum of the execution time of the task and the execution times of all higher priority tasks that are implemented on the same resource. The $\lceil \cdots \rceil$ term determines how often a task $\tilde{t}$ with a higher priority might preempt the execution of task t. The response time r_t exists on both sides of the equation, hence a fix point has to be determined. Starting with $r_t = e_t$, the recurrence equation will reach a fix point if the utilization is below or equals 1.0.

Note that this formulation is only safe as long as the deadline of tasks is lower than their period. If this is not the case, two cases have to be considered. In the first case, the previous instance of a task is dropped when a second instance of the task is ready for execution. Considering the discussion about arbitrary deadlines in [Tindell and Hansson 1995], Equation (1) remains feasible in this case. In the second case, multiple instances of the same task are queued and previous instances have to be considered, assuming a first-come first-served policy. A safe bound that follows [Tindell and Hansson 1995] would extend Equation (1) by adding the term $e_t \cdot \lfloor \tau_t / h_t \rfloor$ to r_t where τ_t is the deadline of the task or application, respectively. This approach has the advantage that it is easy to implement by extending only Constraint (C4) with an additional constant value. Another tighter approach might be to take previous instances into account, considering their actual response time and jitter. This would require the introduction of an additional integer variable to count the actual value of previous instances of the same task $\lfloor (r_t + j_t)/h_t \rfloor$ in Constraint (C4). For further reading on obtaining a tighter bound, we would like to refer the reader to [Lehoczky 1990].

Non-Preemptive Scheduling. Corresponding to the preemptive case, a non-preemptive recurrence equation can be formulated that might be used for instance for the transmission of messages on the CAN bus $a \in A_{\text{CAN}}$. The recurrence equation for this case is presented in [Davis et al. 2007, Eq. (16)] and defined as follows:

$$r_t = e_t + w_t \text{ with } w_t = \max_{\substack{\tilde{t} \in T_a \\ p(\tilde{t}) \geq p(t)}} e_{\tilde{t}} + \sum_{\substack{\tilde{t} \in T_a \\ p(\tilde{t}) < p(t)}} \left\lceil \frac{w_t + j_{\tilde{t}} + \tau_{bit}}{h_{\tilde{t}}} \right\rceil \cdot e_{\tilde{t}} \tag{2}$$

Note that this is a sufficient but not necessary requirement that might lead to some over-approximation which we consider as acceptable. Due to the non-preemptive scheduling, the maximum execution time of any task with a lower priority has to be considered as determined by the $\max$ term. On the other hand, the execution of the considered task cannot be preempted anymore once its execution started. Nevertheless, the task under consideration is suspended until all higher priority tasks are sent. The amount how often a higher priority might suspend the task under consideration is determined by the $\lceil \cdots \rceil$ term where τ_{bit} is the time that the transmission of a single bit requires.

Note that the formulation might not be safe if the deadline is larger than the period and previous instances of the same message have to be considered in case they are queued. An extension can be done correspondingly to the discussed approach for preemptive scheduling, taking previous instances of the same message into account. In the case of a simple over-approximation as discussed in the preemptive case, this would require to extend Constraint (C7).

In a system as illustrated in Figure 5, the response times of all tasks in the task graph have to be calculated to determine whether each deadline τ_t is satisfied. Whether a deadline is violated, is determined by considering all paths to the respective task. Note that for the preemptive and non-preemptive scheduling, higher priority tasks with data-dependencies (there exists a data path between the respective tasks) do not have to be taken into account if the end-to-end deadline of the respective function is lower or equal to their period. This is due to the fact that the data-dependent tasks of the same function can never preempt or suspend the task under consideration. We considered this special case in our implementation without explicitly modeling it in the presented equations.

3. FIXED-PRIORITY SCHEDULING

In the following, the QCQP formulation is presented that is capable of determining a feasible priority assignment for a system, considering all existing deadlines.

3.1. Constraint Encoding

Priority Assignment. In the following, the priority assignment is defined as a set of constraints. For each ECU and CAN bus, the priority function $p : T \to \mathbb{N}$ assigns a unique integer to each task, resulting in a strict order per resource. Thus, the introduced variables and constraints have to model the priority function and the resulting strict order. For this purpose, each binary variable $\mathbf{p}_{\mathbf{t},\tilde{\mathbf{t}}} \in \{0, 1\}$ defines (for the tasks $t, \tilde{t} \in T_a$ with $t \neq \tilde{t}$) whether t has a higher priority than $\tilde{t}$ (1) or not (0). Note that each resource defines its specific $\mathbf{p}_{\mathbf{t},\tilde{\mathbf{t}}}$ variables within its scope which is not further annotated for the sake of simplicity. The constraints for the priority assignment are defined as follows:
$\forall a \in A_{\text{ECU}} \cup A_{\text{CAN}}, t, \tilde{t} \in T_a, t \neq \tilde{t}$:

$$\mathbf{p}_{\mathbf{t},\tilde{\mathbf{t}}} + \mathbf{p}_{\tilde{\mathbf{t}},\mathbf{t}} = 1 \tag{C1}$$

$\forall a \in A_{\text{ECU}} \cup A_{\text{CAN}}, t, \tilde{t}, \hat{t} \in T_a, t \neq \tilde{t}, \tilde{t} \neq \hat{t}, \hat{t} \neq t$:

$$\mathbf{p_{t,\tilde{t}}} + \mathbf{p_{\tilde{t},\hat{t}}} - \mathbf{p_{t,\hat{t}}} \leq 1 \tag{C2}$$

Constraint (C1) satisfies the asymmetry such that if $p(t) < p(\tilde{t})$, then $p(\tilde{t}) < p(t)$ does not hold. At the same time, the constraint ensures comparability. Constraint (C2) models transitivity such that $p(t) < p(\tilde{t})$ and $p(\tilde{t}) < p(\hat{t})$ implies $p(t) < p(\hat{t})$. With the requirement that $t \neq \tilde{t}$ for each $\mathbf{p_{t,\tilde{t}}}$, also irreflexivity is satisfied. Thus, the variables $\mathbf{p_{t,\tilde{t}}}$ model a strict order over the set of priority assignments being irreflexive, asymmetric, and transitive.

We also define a constraint that ensures rate-monotonic scheduling which can be used optionally. This is actually not required but may improve the QCQP solving time significantly as the search space is reduced. The constraint is defined as follows:
$\forall a \in A_{\text{ECU}} \cup A_{\text{CAN}}, t, \tilde{t} \in T_a, h_t < h_{\tilde{t}}$:

$$\mathbf{p_{t,\tilde{t}}} = 1 \tag{C3}$$

Constraint (C3) specifies that a task always has a higher priority if its period is smaller when compared to another task. With this constraint, the optimality might be traded off with a lower runtime. In practice, the optimality is only influenced minimally or not at all like in our case study as discussed later while the runtime of the QCQP is significantly reduced.

Preemptive Scheduling. Given the priority variables, it is possible to determine the response times of tasks. For each task t, the response time is defined as the variable $\mathbf{r_t} \in \mathbb{R}$. Additionally, it is necessary to introduce the release jitter $\mathbf{j_t} \in \mathbb{R}$ for each task. To encode how often a task with a higher priority might preempt the execution, the variable $\mathbf{i_{\tilde{t},t}} \in \mathbb{N}$ is introduced for each pair of tasks $\tilde{t}, t \in T_a$ with $\tilde{t} \neq t$. The response time for a preemptive resource is determined by the following constraints:
$\forall a \in A_{\text{ECU}}, t \in T_a$:

$$\mathbf{r_t} = e_t + \sum_{\tilde{t} \in T_a \setminus \{t\}} \mathbf{p_{\tilde{t},t}} \cdot \mathbf{i_{\tilde{t},t}} \cdot e_{\tilde{t}} \tag{C4}$$

$\forall a \in A_{\text{ECU}}, t, \tilde{t} \in T_a, t \neq \tilde{t}$:

$$\mathbf{i_{\tilde{t},t}} \geq \frac{1}{h_{\tilde{t}}} \cdot \mathbf{r_t} + \frac{1}{h_{\tilde{t}}} \cdot \mathbf{j_{\tilde{t}}} \tag{C5}$$

Constraint (C4) corresponds to Equation (1) where the variable $\mathbf{p_{\tilde{t},t}}$ defines whether some other task has a higher priority and $\mathbf{i_{\tilde{t},t}}$ corresponds to the amount of preemptions that are possible by the other task. The value of $\mathbf{i_{\tilde{t},t}}$ is determined by Constraint (C5). Note that the variable for the number of preemptions is merely a lower bound such that the response time might be over-approximated using these constraints. However, the solver will ensure that the over-approximation is never higher than necessary to fulfill all deadline constraints if a feasible solution exists.

Non-Preemptive Scheduling. Corresponding to the preemptive schedule, the constraints for the non-preemptive scheduling are defined in the following. The formulation requires the additional variable $\mathbf{b_t} \in \mathbb{R}$ that encodes the maximum delay by a lower priority task. The constraints are formulated as follows:
$\forall a \in A_{\text{CAN}}, t \in T_a$:

$$\mathbf{r_t} = e_t + \mathbf{w_t} \tag{C6}$$

$$\mathbf{w_t} = \mathbf{b_t} + \sum_{\tilde{t} \in T_a \setminus \{t\}} \mathbf{p_{\tilde{t},t}} \cdot \mathbf{i_{\tilde{t},t}} \cdot e_{\tilde{t}} \tag{C7}$$

$$\mathbf{b_t} \geq e_t \tag{C8}$$

$\forall a \in A_{\text{CAN}}, t, \tilde{t} \in T_a, t \neq \tilde{t}$:

$$\mathbf{b_t} \geq \mathbf{p_{t,\tilde{t}}} \cdot e_{\tilde{t}} \tag{C9}$$

$$\mathbf{i_{\tilde{t},t}} \geq \frac{1}{h_{\tilde{t}}} \cdot \mathbf{w_t} + \frac{1}{h_{\tilde{t}}} \cdot \mathbf{j_{\tilde{t}}} + \frac{\tau_{bit}}{h_{\tilde{t}}} \tag{C10}$$

Constraint (C6) and (C7) correspond to Equation (2). Constraints (C8) and (C9), respectively, determine the lower bound for the maximum delay due to a previous instance of the task itself or a task with lower priority. Constraint (C10) determines how often a higher priority task might suspend the considered task. Corresponding to the preemptive scheduling, the constraints define a lower bound for the worst-case response time determination which is sufficient to obtain a feasible schedule.

Fixed-Delay Scheduling. For components like gateways, in many cases the worst-case time is specified in a data-sheet. In this case, the response time is determined as follows:

$\forall a \in A_{\text{GW}}, t \in T_a$:

$$\mathbf{r_t} = r_{\text{GW}} \tag{C11}$$

Constraint (C11) sets the response time of tasks to a predefined worst-case value.

End-to-end latencies. In order to determine whether each deadline is satisfied, it is necessary to calculate the delay and release jitter for each task along each path in the task graph. Note that the proposed formulation has to consider different activation patterns for tasks and messages, respectively. Corresponding to [Zheng et al. 2007], the *periodic* and *event-driven* activation model can be applied. The following equations specifically model the *event-driven* activation where tasks are activated once the input data becomes available. In order to adapt the equations towards *periodic* activation, it is necessary to add the task period on the right-hand-side of Constraint (C14). Whether tasks of a function follow a *periodic* or *event-driven* activation model is a trade-off between timeliness and composability as discussed in [Matic and Henzinger 2005].

For each task t, the delay is defined as a variable $\mathbf{d_t} \in \mathbb{R}$ and the release jitter is defined as $\mathbf{j_t} \in \mathbb{R}$. The constraints to determine the values of these variables and to determine whether the deadlines are fulfilled are as follows:

$\forall t \in T, \neg\exists(\tilde{t}, t) \in E_T$:

$$\mathbf{d_t} = \mathbf{r_t} \tag{C12}$$

$$\mathbf{j_t} = 0 \tag{C13}$$

$\forall t \in T, (\tilde{t}, t) \in E_T$:

$$\mathbf{d_t} \geq \mathbf{r_t} + \mathbf{d_{\tilde{t}}} \tag{C14}$$

$$\mathbf{j_t} \geq \mathbf{r_{\tilde{t}}} + \mathbf{j_{\tilde{t}}} - e_{\tilde{t}}^{\min} \tag{C15}$$

$\forall t \in T$:

$$\mathbf{d_t} \leq \tau_t \tag{C16}$$

For each task that has no predecessor, the delay and jitter are defined by the Constraints (C12) and (C13), respectively. Here, the initial release jitter is assumed to be 0. It might be defined as any value other than 0 if specified by the respective sensor or sender. In case a task has predecessors, the delay and jitter values are determined by the Constraints (C14) and (C15), respectively. The delay is determined as the sum of the maximum of the delays of the preceding tasks and the response time of the current task. Correspondingly, the jitter is the maximum of the preceding task jitter values and the current jitter which is the difference between the worst-case response time and the best-case execution time ($e_{\tilde{t}}^{\min}$) which might be set to 0 if it is not known. Finally, Constraint (C16) ensures that all deadlines are satisfied by constraining the delays.

3.2. Priority Value Determination

In contrast to Integer Linear Programs (ILPs), a QCQP might also contain terms that are quadratic, considering products of two variables. This is required here in Constraints (C4) and (C7) that model Equations (1) and (2), respectively. In case the QCQP is convex (as it is the case in the provided formulation), modern solvers [Gurobi Optimization, Inc. 2015] can determine a feasible priority assignment efficiently, using an interior point method.

The full QCQP that determines fixed priorities is defined as follows:

$$\begin{aligned} &\text{maximize} && \sum_{\substack{t \in T \\ \neg \exists \tilde{t} \in T : (t,\tilde{t}) \in E_T}} \tau_t - \mathbf{d_t} \\ &\text{subject to} && \\ &\quad \text{Constraints (C1) – (C2)} && \\ &\quad \text{Constraint (C3) } \textit{optional} && \\ &\quad \text{Constraints (C4) – (C16)} && \end{aligned} \tag{QCQP1}$$

The objective function maximizes the sum of slacks of all functions by considering all tasks that do not have any successors and a known deadline. Since this is at the same time maximizing the average slack, on average the priority bounds can be determined more flexibly in the next step. Alternatively, it would be possible to optimize the smallest slack of all tasks by introducing an additional variable that bounds the minimal slack of all tasks and maximizing this variable. Since the experimental results consider the average diversity, the objective function in (QCQP1) yields better solutions than maximizing the minimal slack which, in turn, does not consider tasks with a relatively large slack. Note that the rate monotonic requirement in Constraint (C3) is optional. In practice, often optimal solutions enforce a rate monotonic scheduling and in this case, the runtime can be significantly improved. For instance, the rate monotonic constraint has no influence on the results of our case study while reducing the runtime from more than one hour to $9.0\,\mathrm{s}$.

For the example in Figure 5, the QCQP determines the solution that corresponds to the values in Figure 6 that shows the obtained response times, end-to-end delays, and priorities or identifiers, respectively. The priority function is determined by the solution values of the $\mathbf{p_{t,\tilde{t}}}$ variables that define a strict order. The order is defined by

$$p(t) < p(\tilde{t}) \Leftrightarrow \mathbf{p_{t,\tilde{t}}} = 1. \tag{3}$$

Given the order, the priorities can be assigned accordingly, starting from 1 with the highest priority task to n for the n-th message. Accordingly, priorities for the ECUs can be determined.

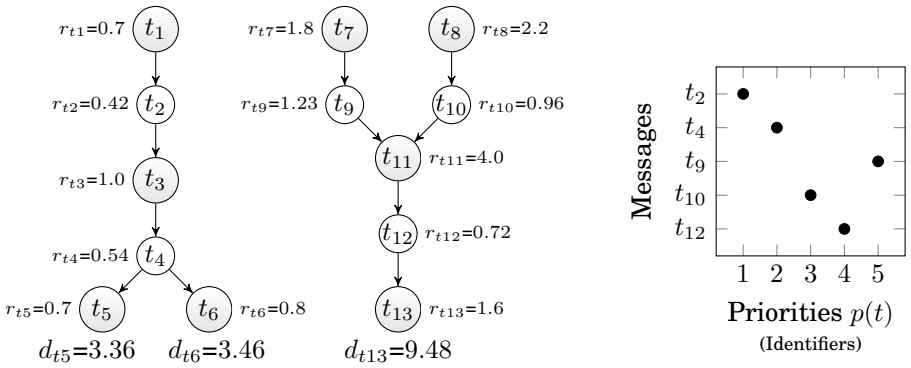

Fig. 6. Task graph that corresponds to the feasible priority assignment for the system in Figure 5 with the priorities for messages. Annotated are the task response times and the end-to-end latencies. The priorities are as follows for ECU_1 $p(t_5) < p(t_1) < p(t_{11})$, for ECU_2 $p(t_3) < p(t_7)$, and for ECU_3 $p(t_6) < p(t_8) < p(t_{13})$.

4. BOUNDED-PRIORITY SCHEDULING

In the following, the QCQP formulation is proposed that determines bounds for the priorities of CAN messages. The approach uses the fixed priorities from the previous section to determine these bounds.

4.1. Constraint Encoding

Corresponding to the function p, a priority set function $P : T \to 2^{\mathbb{N}}$ is defined. This set determines all possible priorities of a task and is constrained by a lower bound and upper bound, respectively. The set is defined such that if tasks are assigned any unique priority per resource within their priority bounds, the end-to-end latencies are still satisfied.

If bounded priorities are used, Equation (2) has to be adapted correspondingly. Here, we take advantage of the monotonicity of the determination of worst-case response times such that only the maximum amount of tasks has to be considered that might preempt or suspend the task under consideration. This maximum amount of messages is determined by considering the upper bounds and lower bounds of priorities, respectively, as shown in the following.

$$r_t = e_t + w_t \text{ with } w_t = \max_{\substack{\tilde{t} \in T_a, \tilde{t} \neq t \\ \max\{P(\tilde{t})\} \geq \min\{P(t)\}}} e_{\tilde{t}} + \sum_{\substack{\tilde{t} \in T_a, \tilde{t} \neq t \\ \min\{P(\tilde{t})\} \leq \max\{P(t)\}}} \left\lceil \frac{w_t + j_{\tilde{t}} + \tau_{bit}}{h_{\tilde{t}}} \right\rceil \cdot e_{\tilde{t}} \quad (4)$$

That means a message under consideration t might be delayed by any message $\tilde{t}$ that has a smaller or equal lower priority bound compared to the upper priority bound of t. Thus, there exists at least one scenario when $\tilde{t}$ might delay t. Additionally, the $\max$ term has to be adapted such that all other messages that might in any scenario have a lower priority (given by a larger or equal identifier) have to be considered.

For CAN buses, Equation (2) is superseded by Equation (4) in the following. However, it is not required to adapt Constraints (C6) to (C10) in any way as these already implicitly have the capability of modeling Equation (4). Instead, Equation (4) is modelled by adapting constraints on the priority variables $\mathbf{p_{t,\tilde{t}}}$ that are part of the non-preemptive scheduling constraints.

The first additional constraint has to ensure that all existing priorities are considered as follows:

$\forall a \in A_{\text{ECU}} \cup A_{\text{CAN}}, t, \tilde{t} \in T_a, p(t) < p(\tilde{t})$:

$$\mathbf{p_{t,\tilde{t}}} = 1 \tag{C17}$$

In combination with the asymmetry Constraint (C1), this defines the fixed priority function for tasks on ECUs. Correspondingly, Constraints (C1) and (C2) are omitted for CAN buses such that a message might have multiple priority values and, in result, defined bounds. In this case, the variable $\mathbf{p_{t,\tilde{t}}} \in \{0, 1\}$ is redefined and states whether a task t might have a higher priority than task $\tilde{t}$.

To optimize the priority bounds, additional binary variables $\mathbf{s_{t,\tilde{t}}} \in \{0, 1\}$ are introduced for messages $t, \tilde{t} \in T_a$ that are routed on the same CAN bus $a \in A_{\text{CAN}}$. These additional variables are only introduced for messages that satisfy $p(t) < p(\tilde{t})$. The variable $\mathbf{s_{t,\tilde{t}}}$ indicates whether the priority of task t might also be lower than the priority of task $\tilde{t}$ when it evaluates to 1. This *shifting* of priorities is expressed by the following constraint:
$\forall a \in A_{\text{CAN}}, t, \hat{t}, \tilde{t} \in T_a, p(t) < p(\hat{t}) \leq p(\tilde{t})$:

$$\mathbf{p_{\hat{t},t}} - \mathbf{s_{t,\tilde{t}}} \geq 0 \tag{C18}$$

Constraint (C18) specifies that if message t is decreased in its priority such that message $\tilde{t}$ has a higher priority, all messages $\hat{t}$ that have their priorities between t and $\tilde{t}$ may now suspend message t. Correspondingly, the binary variables $\mathbf{p_{\hat{t},t}}$ evaluate to 1.

4.2. Priority Bound Determination

With the additional constraints, the QCQP is defined as follows:

$$\begin{aligned}
&\text{maximize} \sum_{\substack{a \in A_{\text{CAN}}, t, \tilde{t} \in T_a \\ p(t) < p(\tilde{t})}} \mathbf{s_{t,\tilde{t}}} \\
&\text{subject to} \\
&\qquad \text{Constraints (C1) – (C2) for } a \in A_{\text{ECU}} \\
&\qquad \text{Constraint (C3) } \textit{optional} \\
&\qquad \text{Constraints (C4) – (C16)} \\
&\qquad \text{Constraints (C17) – (C18)}
\end{aligned} \tag{QCQP2}$$

The objective function maximizes the amount of priority shifting of tasks and, in result, the average bound span over all messages. From the result of the QCQP, it is possible to determine the priority set $P(t)$ for each message. This is done by determining the upper and lower bound of the set, respectively.

For the bus $a \in A_{\text{CAN}}$, the upper bound of each task $t \in T_a$ is determined as follows:

$$\max\{P(t)\} = \max_{\substack{\tilde{t} \in T_a: \\ \mathbf{s_{t,\tilde{t}}} = \mathbf{1}}} \{p(\tilde{t})\} \tag{5}$$

Here, the maximum priority shift is determined that represents the upper bound. Determining the lower bound requires to consider all lower priority messages that can be shifted beyond the task under consideration. For the bus $a \in A_{\text{CAN}}$, the lower bound of each task $t \in T_a$ is determined as follows:

$$\min\{P(t)\} = \min_{\substack{\hat{t}, \tilde{t} \in T_a, p(\tilde{t}) \leq p(\hat{t}) < p(t): \\ \mathbf{s_{\hat{t},t}} = \mathbf{1}}} \{p(\tilde{t})\} \tag{6}$$

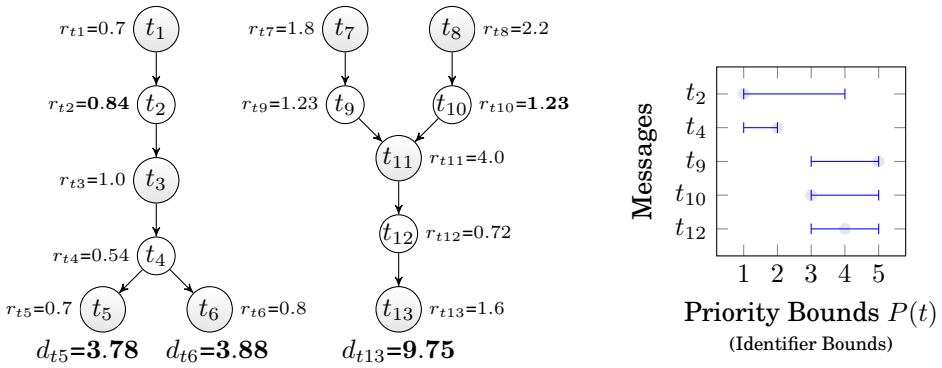

Fig. 7. Illustration of the task graph that corresponds to the feasible priority bound assignment for the system in Figure 5, using the fixed priorities from the solution as given in Figure 6. Annotated are the task response times and the end-to-end latencies. Those values that increased in comparison to the fixed priority scheduling are bold. The priorities are as follows for ECU_1 $p(t_5) < p(t_1) < p(t_{11})$, for ECU_2 $p(t_3) < p(t_7)$, and for ECU_3 $p(t_6) < p(t_8) < p(t_{13})$.

Here, the lower bound is defined as the priority of the task that can switch its priority with the task under consideration. However, also all other tasks that have their original priorities between these two tasks have to be capable of switching the priority with the task under consideration.

Figure 7 shows the determined priority bounds for the example from Figure 5 and the fixed priorities from Figure 6. The determined worst-case response times are increased for all messages. The results are based on the following assignment of $\mathbf{p}_{\mathbf{t},\tilde{\mathbf{t}}}$ variables:

	t_2	t_4	t_{10}	t_{12}	t_9
t_2		1	1	1	1
t_4	1		1	1	1
t_{10}	1	0		1	1
t_{12}	1	0	1		1
t_9	0	0	1	1	

The grayed out 1 values in the upper right triangle are already predetermined by the results of (QCQP1) which are set in Constraint (C17). With the remaining values in the matrix, the shifting is determined with Equations (5) and (6), resulting in the priority bounds in Figure 7. Since the upper bounds of the priorities of t_4, t_{10}, and t_{12} are larger than the fixed priorities, the end-to-end latencies of both functions are in result increased while still satisfying all deadlines.

5. PRIORITY/IDENTIFIER SAMPLING

With the determined bounds, it becomes necessary to sample different priorities or identifiers, respectively, for each vehicle in a final step. We propose a Monte Carlo method that is illustrated in Figure 8 and briefly summarized in the following. The method is applied to each CAN bus separately in a system with multiple buses that are connected via one or more gateways. First, for each message a priority is sampled within the defined bounds with a real value $x_t \in [\min\{P(t)\}, \max\{P(t)\}]$. In a second step, the values are discretized by sorting the priorities by the sampled real values such that the following equation is satisfied:

$$p'(t) < p'(\tilde{t}) \Leftrightarrow x_t < x_{\tilde{t}} \tag{7}$$

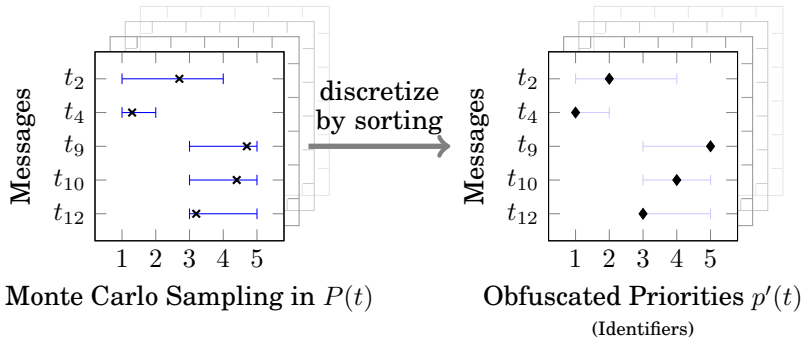

Fig. 8. The general priority sampling. First real values are sampled within the determined priority bounds and subsequently these values are sorted to determine the actual priorities or identifiers, respectively.

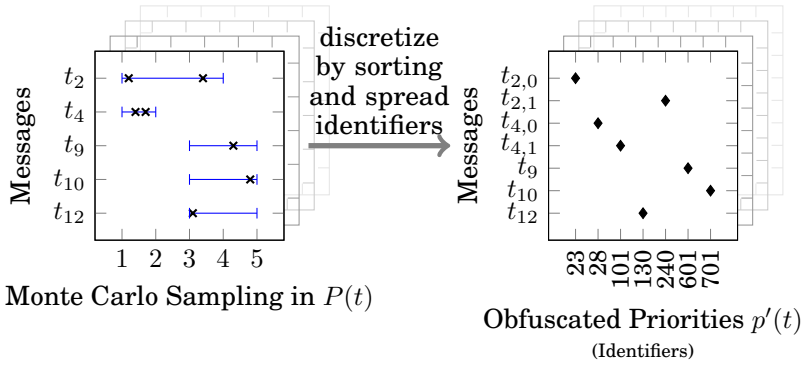

Fig. 9. Illustration of an enhanced obfuscation priority generation that (1) splits messages into multiple instances to obfuscate their periods and (2) uses the full range of priorities for the assignment. The first step prevents a simple categorization of messages by their period (here all seven messages have now the same period of $10\,\mathrm{ms}$). The second step further reduces the probability that two messages have the same identifiers for two distinct obfuscated priority assignments.

Here, p' represents the obfuscated priorities or identifiers, respectively. This approach is very fast as it only requires a sampling step once for each message and a final sorting that can be implemented very efficiently. Thus, the complexity is within $\mathcal{O}(n \log n)$ with n being the number of messages. This polynomial time approach is significantly faster than the previous two stages that relied on mathematical solvers which have an exponential worst-case complexity.

The proposed sampling might be extended by further measures to increase the obfuscation of CAN messages which are outlined in the following and illustrated in Figure 9. This is particularly important to prevent system identification methods like [Di Natale and Zeng 2010] to classify messages based on their periods. A single message stream that is transmitted with a period h might be split into m message streams with periods $h \cdot m$. Note that these message streams have a mutual offset of h. This might be useful for our example from Figure 5 where the first function has a period of $5\,\mathrm{ms}$ and the second function a period of $10\,\mathrm{ms}$. A compromised device inside the vehicle might quickly categorize these by profiling the bus for a short time. In our example, the messages t_2 and t_4 might be split into two messages $t_{2,0}$, $t_{2,1}$ and $t_{4,0}$, $t_{4,1}$, respectively. If these messages are sent alternatingly, an observer perceives

for the entire system seven messages with a period of $10\,\mathrm{ms}$. Further profiling might reveal offsets and information about the relation of these messages but this requires additional time and computational power which are both constrained in an automotive attack scenario. In summary, the optimal solution would be to identify the least common multiple of all message periods on the bus and use this value for the message splitting.

Practically, the sampling with message splitting is implemented correspondingly to the above mentioned sampling approach, given that enough identifiers exist. Here, a set of random variables $X_t \in [\min\{P(t)\}, \max\{P(t)\}]$ for each message is determined. In the next step, these values are sorted to determine a priority order for the obfuscated priority assignment.

Finally, the full range of identifiers might be used to reduce the probability that two messages have the same identifiers for two distinct obfuscated priority assignments. For standard CAN, the priorities can be encoded by $11\,\mathrm{bit}$, resulting in 2048 different identifiers. For extended CAN, this number is significantly higher since the identifier field has $29\,\mathrm{bit}$. Such a randomization is a further step that is implemented with an additional sampling to the given full range of identifiers, considering already sampled values that form lower and upper bounds, respectively.

6. EXPERIMENTAL RESULTS

In the following, experimental results are presented that give evidence of the applicability of the proposed approach. For this purpose, a set of synthetic test cases is evaluated and a large realistic case study from the automotive domain is introduced. All experiments have been carried out on an Intel Xeon QuadCore CPU with 3.20 GHz and 12 GB RAM using the GUROBI QCQP solver [Gurobi Optimization, Inc. 2015].

6.1. Synthetic Test Cases

In order to show the scalability of the proposed approach and its properties in terms of priority obfuscation, a set of 500 synthetic test cases is used. The test cases have been randomly generated and have various complexities. The smallest test case consists of 2 ECUs, 1 CAN bus, and 2 functions with 11 tasks. The largest test case consists of 23 ECUs, 4 CAN buses connected via a central gateway, and 25 functions resulting in 294 tasks. For all test cases, deadlines have been randomly chosen to vary the possible slack of tasks[2].

First, the runtime of the proposed methodology is investigated. The runtimes of the fixed-priority scheduling (QCQP1) and bounded-priority scheduling (QCQP2) for all test cases are illustrated in Figure 10. Note that we use the optional Constraint (C3) for rate-monotonic scheduling for the fixed-priority scheduling while it is omitted for the bounded-priority scheduling. This ensures an efficient solving of (QCQP1) while it increases the bound span for (QCQP2).

It can be observed that most of the test cases can be optimally solved within a few seconds. For some larger test cases, the timeout of $3600\,\mathrm{s}$ is reached. This is the case for 6 test cases for (QCQP1) and 51 test cases for (QCQP2), which translates to around 1% of the test cases for the fixed-priority scheduling and around 10% for the bounded-priority scheduling. This shows that the approach is very robust and capable of obtaining optimal solutions for most problems of our synthetic test cases. At the same time, for all test cases that cannot be solved optimally within the given time limit, the QCQP solver is capable of providing an *optimized* solution. Such an optimized solution might already be sufficient as trade-off to the solver runtime. In this case, the optimality gap is a good indicator determined by $(\mathrm{ObjBound} - \mathrm{ObjVal})/|\mathrm{ObjVal}|$ where $\mathrm{ObjBound}$ and ObjVal are the objective bound and best solution objective, respectively, see [Gurobi Optimization,

[2]All terminal tasks have a defined deadline.

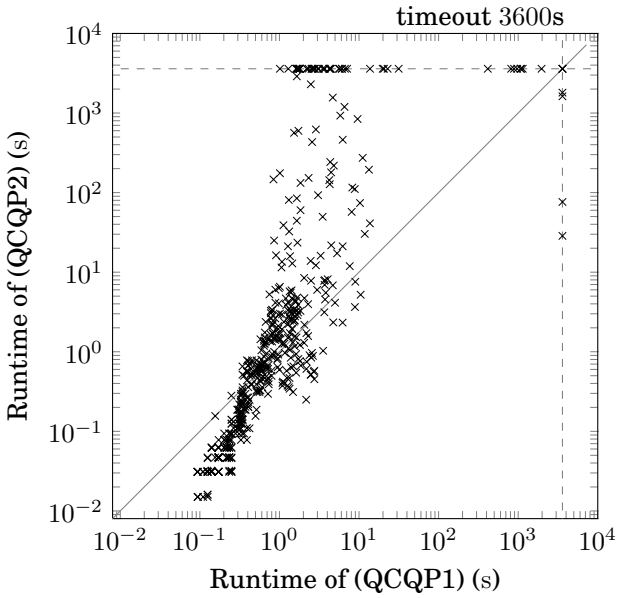

Fig. 10. Illustration of the QCQP runtimes for 500 synthetic test cases. Each point shows the runtime for one test case for fixed-priority scheduling (QCQP1) and bounded-priority scheduling (QCQP2), respectively. A timeout of 3600 s for both scheduling approaches is chosen.

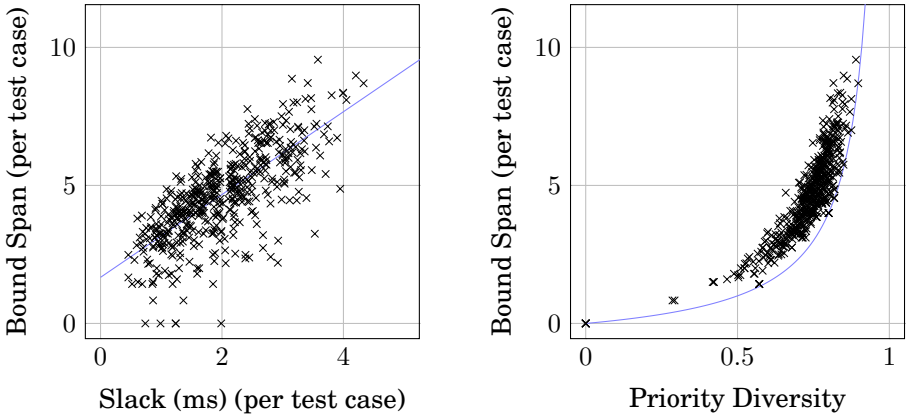

Fig. 11. Illustration of the relation between the average deadline slack and resulting bound span (left plot) and the relation between the bound span and resulting priority diversity (right plot) for the 500 synthetic test cases. The line on the left plot illustrates the linear regression while the curve on the right plot is the inverse of $x/x+1$ which equals the probability that two sampled integers from the bound span are equal.

Inc. 2015]. Note that the optimality gap is always between 0 and 1 while the optimal value equals to 0. For (QCQP1), the largest optimality gap is 0.0027 and for (QCQP2) it is 0.0034. This shows that the optimized solutions are very close to optimality.

Figure 11 illustrates the relation between slack, bound span, and priority diversity. The slack is defined for tasks $t \in T$ that have a specified deadline τ_t as follows:

$$\tau_t - \mathbf{d_t} \tag{8}$$

Table I. Summary of the clusters, ECUs, number of tasks, and periods of the functions of the case study. The number of tasks is summarized from the tasks that are implemented on ECUs as well as routed via CAN buses and gateways.

Cluster (CAN bus)	**ECUs**	**Tasks** (ECU+CAN+GW)	**Functions / Periods** (ms)
Body (CAN_b)	6	71 $(34 + 28 + 9)$	$40, 40, 50, 80, 80, 80$
Driver Assistance (CAN_d)	3	74 $(31 + 31 + 12)$	$5, 10, 20, 40, 80, 80$
Chassis (CAN_c)	4	97 $(36 + 43 + 18)$	$10, 20, 40, 40, 50, 50, 100$
Infotainment (CAN_i)	4	38 $(16 + 16 + 6)$	$5, 80, 100$
Powertrain (CAN_a)	8	52 $(32 + 19 + 1)$	$5, 10, 40, 50, 50, 80$
System	25	332 $(149 + 137 + 46)$	

Note that the objective function of (QCQP1) is maximizing the slack. This slack has a direct relation to the bound span of a message t that is obtained from (QCQP2) and defined as follows:

$$\max\{P(t)\} - \min\{P(t)\} \tag{9}$$

It can be seen in Figure 11 that a larger slack after the first stage generally leads to a larger bound span after the second stage. Note that the values for the slack and bound span are calculated as average per test case. This shows that a maximization of the slack in the first stage is a good choice as a larger bound span is desirable towards a better obfuscation as shown in the following.

The obfuscation quality is determined by sampling a sufficiently large set of priority assignments in stage three and comparing these mutually. We sampled a set of 1000 priority assignments for each test case while the sampling in each case always required less than $1\,\text{ms}$. The obfuscation quality is measured by a diversity value that is determined for a task $t \in T$ of two priority assignments as follows:

$$|\text{sgn}(p'(t) - p''(t))| \tag{10}$$

Here, p' and p'', respectively, denote the priorities in the specific assignment. Note that we used the sampling method as illustrated in Figure 8 such that the resulting value from Equation (10) is meaningful. Considering the average over all tasks and priority assignments per test case, a higher value indicates a higher diversity and, thus, a better obfuscation quality. It can be observed that a larger bound span clearly increases the obfuscation quality.

6.2. Case Study

In order to give evidence of the applicability of the proposed methodology for realistic problems, the case study from [Lukasiewycz et al. 2013] is used. The case study consists of five CAN buses that define clusters and a central gateway that interconnects these clusters. Note that the functions in each cluster are not solely mapped to ECUs of this cluster such that it is not possible to determine the priorities for each cluster separately. The details of the case study are summarized in Table I which also illustrates the periods of the functions in each cluster. Overall, the case study consists of 25 ECUs and 27 functions that result in 332 tasks. In summary, 149 tasks are implemented on the ECUs that require a feasible priority assignment while 137 priorities have to be assigned to the existing message tasks that are routed via CAN buses. Note that these 137 message tasks are obtained from 88 message streams from which some are routed via the gateway. The number of message tasks that are routed via the gateway is 46 and a fixed worst-case delay of $1\,\text{ms}$ in the gateway is assumed. The utilization with respect to the considered functions for the buses CAN_b, CAN_d, CAN_c, CAN_i, and CAN_a is 20.6%, 24.5%, 16.6%, 22.1%, and 33.8%, respectively. Note that the actual utilization is higher due to low priority messages that have very large deadlines and these messages

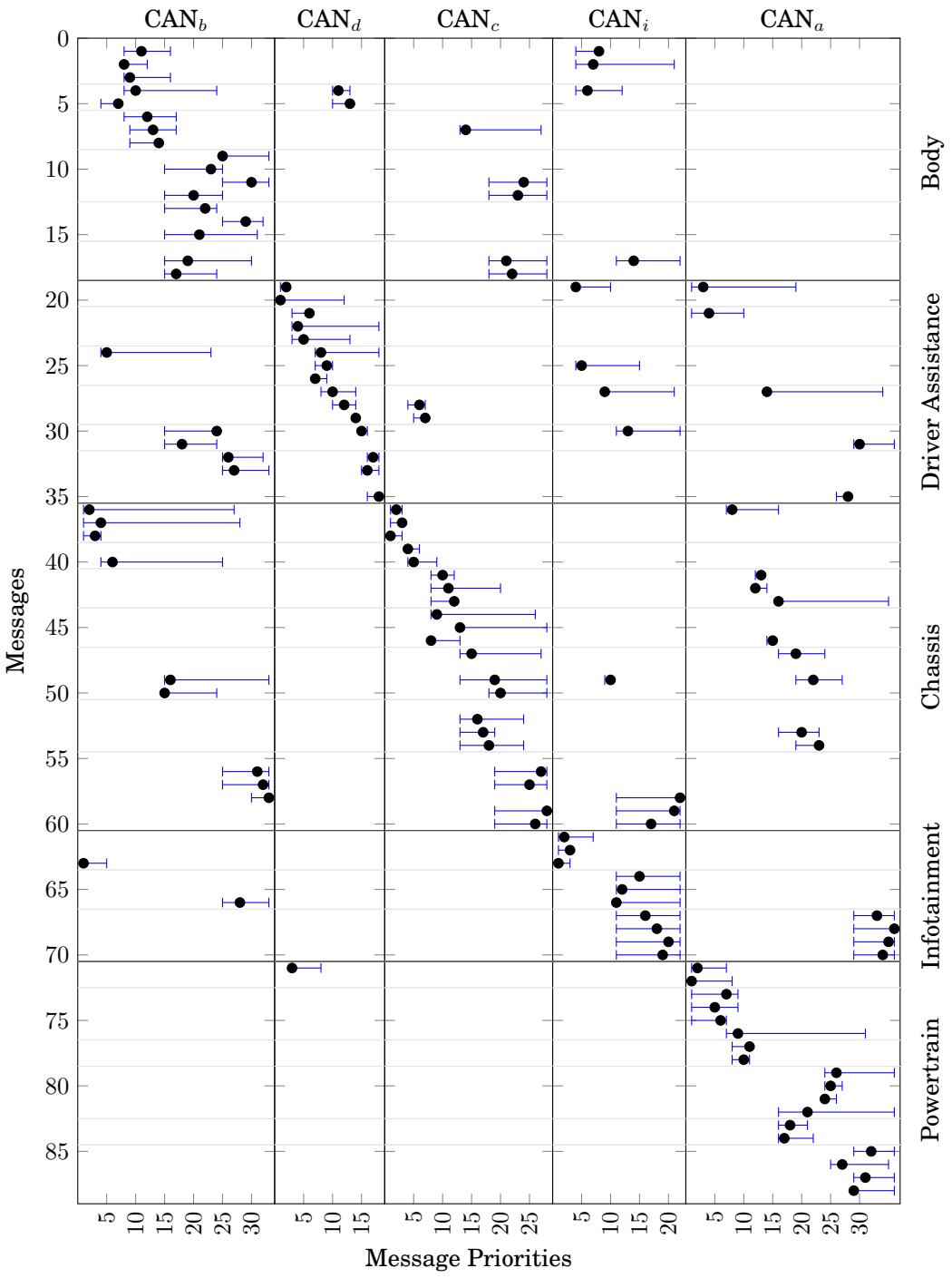

Fig. 12. Illustration of the fixed priorities (●) and bounded priorities (⊢⊣) for 88 message streams on 5 CAN buses in the proposed case study. The messages are ordered by clusters and functions.

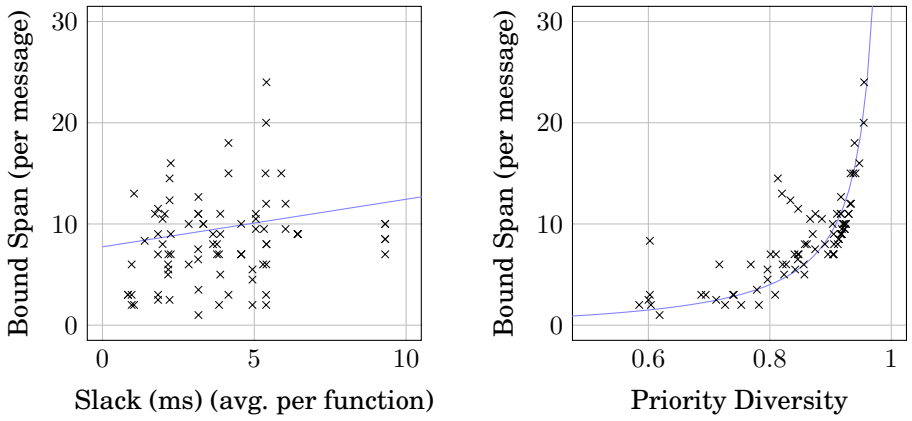

Fig. 13. Illustration of the relation between the average deadline slack and resulting bound span (left plot) and the relation between the bound span and resulting priority diversity (right plot) for the case study. The line on the left plot illustrates the linear regression while the curve on the right plot is the inverse of $x/x+1$ which equals the probability that two sampled integers from the bound span are equal.

are, therefore, not considered. In summary, the case study has tight function slacks in relation to the periods which are shown in Table I, respectively.

With the proposed methodology, it is possible to determine a feasible fixed-priority assignment within $8.1\,\mathrm{s}$ using (QCQP1). The runtime of (QCQP2) exceeded the timeout of $3600\,\mathrm{s}$ with a very small optimality gap of 0.0034. Although the optimization is aborted due to the timeout, the resulting average bound span is relatively high with a value of 9.36. A detailed illustration of all fixed priorities and bounded priorities for the case study are illustrated in Figure 12. With an average bound span of 9.36 and 137 distinct priorities for message streams, it is possible to approximate the number of distinct priority assignments to $9.36^{137} = 1.16 \cdot 10^{133}$. This value shows that the probability that two vehicles from an entire fleet have the exactly same priority assignment is negligibly small. For a certain function with x messages, the number of distinct priorities can be correspondingly approximated by 9.36^x. For instance, in the case of four messages, the probability that two vehicles have the identical priority assignment is around 0.01%. Note that with the techniques of message splitting and random sampling of identifiers as illustrated in Figure 9, the priorities are further obfuscated.

Figure 13 illustrates the relation between slack, bound span, and priority diversity for the case study. The results are visualized correspondingly to the synthetic test cases except that the average is calculated for each message while the slack is determined for the function of the considered message. These results show again a relation between the slack and bound span as well as the bound span and priority diversity. Note that the sampling of priorities for this large test case never required more than $1\,\mathrm{ms}$.

7. RELATED WORK

This section gives an overview of related work in the domain of systems with fixed-priority scheduling and the related priority assignment problem. Additionally, we discuss related work in the field of automotive security.

7.1. Fixed-Priority Scheduling

A real-time analysis for priority-based schedulers with predefined priorities might be done efficiently using response-time analysis. A general approach for preemptive systems is presented in [Joseph and Pandya 1986] which might be applied to ECUs

with OSEK or AUTOSAR OS schedulers. For non-preemptive systems like the CAN bus, the method is adapted in [Tindell et al. 1995] and later revised in [Davis et al. 2007]. Both approaches are applied in our paper and explained in Section 2. In general, the priorities of these systems are defined in the integration phase in order to satisfy all deadlines.

General priority optimization of messages and tasks, respectively, can be done with heuristics or exact approaches using mathematical programming solvers. Meta-heuristic approaches such as presented in [Hamann et al. 2004] rely on Evolutionary Algorithms (EAs) or Simulated Annealing (SA) to optimize the priorities of messages and tasks. The randomness of these approaches might be used to obtain obfuscated priority assignments. However, in practice these heuristic approaches do not perform well when the deadline constraints are tight and at the same time the runtime becomes unacceptably large.

Other approaches to determine task and message priorities as presented in [Metzner and Herde 2006; Reimann et al. 2010; Reimann et al. 2011] are applied within an architectural decision process and rely on iterative pruning. In each iteration, a priority assignment is determined and analyzed such that in case of violated deadline constraints, a part of the search space is pruned. However, as discussed in [Lukasiewycz et al. 2013], pruning infeasible solutions does not scale well and is, therefore, not applicable to large and realistic problems.

An approach to tackle the priority assignment problem with an ILP is presented in [Lisper and Mellgren 2001] where applications cannot be defined in general but are restricted to chains of tasks and the problem has to be linearized, introducing a high number of additional constraints and variables. This approach was extended towards QCQP in [Lukasiewycz et al. 2013], showing a compact problem representation and very good scalability. In this paper, we have extended [Lukasiewycz et al. 2013] for the first stage by avoiding an intermediate graph representation. As a result, the formulation becomes even easier to understand, implement, and as a result more extensible.

All mentioned approaches focus merely on the task of obtaining a single feasible fixed-priority assignment as it is necessary in our first step (QCQP1). While our formulation in the first step improves the state-of-the-art in [Lukasiewycz et al. 2013], stage two and three, that provide the priority bounds and obfuscation by sampling, are to the best of our knowledge not considered by any related work. Thus, the correlation between existing message priorities and scaling effects on vehicle fleets are for the first time considered in the work at hand.

Finally, it should be mentioned that obfuscation techniques might be also combined with other aspects of scheduling. For instance, an approach like [Davare et al. 2007] that optimizes the periods might be used to obtain systems with different periods of messages. By considering work like [Zhu et al. 2012] that concurrently optimizes task priorities and task distribution, the obfuscation might also involve task mapping aspects.

7.2. Automotive Security

Several approaches in the context of automotive security have been published in literature. In [Koscher et al. 2010; Checkoway et al. 2011], an undisclosed vehicle has been analyzed for its security properties where scaling effects in vehicle fleets are mentioned as a major concern. Due to the similar nature of vehicle models, exploits and viruses can easily affect entire fleets. More recently, in [Miller and Valasek 2013; 2014] in-vehicle networks have been categorized in terms of security based on their CAN structure. With many internal communication systems being increasingly connected to external networks via wireless connections, the risk of access by unauthorized parties is increasing significantly. To counter vehicle attacks, an intrusion detection system is

proposed. A recent overview of security challenges is given in [Sagstetter et al. 2013] where security threats are classified by attack vector and application domain.

Multiple different protection mechanisms have been proposed for automotive networks and other resource-constrained embedded communication systems. In [Zalman and Mayer 2014], Message Authentication Codes (MACs) are introduced to ensure end-to-end protection of messages against alteration and replay attacks. MACs are integrated with error correction of conventional communication systems such that the overhead is minimized. In [Lin et al. 2013], message security through MACs is combined with a Mixed Integer Linear Programming (MILP) to schedule messages efficiently on a CAN bus with minimal overhead. The MILP is further modified in [Lin et al. 2014] to reduce the computational complexity. In [Han et al. 2014], a time-released key system is employed to secure time-triggered FlexRay communication. A MAC is used to protect messages, ensuring integrity and securing against replay attacks. Messages are scheduled efficiently with a MILP, considering the added security requirements. In [Mundhenk et al. 2015] an authentication mechanism for ECUs is presented, allowing fast and efficient authentication of ECUs and authorization of message streams.

The abovementioned approaches utilize MACs to ensure message integrity or focus on authentication and authorization for bus integrity. In this paper, we do not consider protection in the classic security categories confidentiality, integrity and availability. Instead, we target scaling effects on vehicle fleets. Due to the large number of identical vehicles, an attack developed on one vehicle instance might easily scale to the complete fleet of vehicles. Our proposed approach allows to find multiple solutions to a given CAN message priority assignment problem which can be employed across the vehicle fleet. Thus, we mitigate the impact of any broken security measure as any attack can only be applied to a very limited number of vehicles. Our approach can flexibly be extended with security mechanisms as proposed in literature. This allows to protect confidentiality, integrity and availability while mitigating scaling effects.

Exemplary, in [Woo et al. 2014], an attack on an after-market OBD adapter via a malicious smartphone application is described. These adapters offer diagnosis and logging functions for drivers are for instance connected via Bluetooth. By bridging these external connections to the OBD port, a significant attack vector is created. As these devices are sold at low prices in high volumes, the impact across vehicles is significant. By running one attack against multiple vehicles of one model or manufacturer, considerable damage can be achieved. With our proposed approach such an attack does not scale as every vehicle is using obfuscated CAN identifiers and needs to be analyzed (and attacked) separately to create significant impact.

8. CONCLUDING REMARKS

This paper presents an approach to determine obfuscated identifiers for CAN platforms, considering real-time constraints. Orthogonally to encryption and firewalls, obfuscated priorities increase the security of CAN buses by mitigating scaling effects. This is particularly important in automotive applications where cost-effective security approaches are necessary. The proposed approach is using QCQP to first determine fixed priorities which are subsequently extended to priority bounds. While this bound determination is computationally expensive, it is required only once in the design process. In order to determine a specific priority assignment by sampling within the bounded priorities, a very efficient approach is introduced that for all test cases never requires more than $1\,\mathrm{ms}$ per vehicle. Our test cases and case study show that CAN identifiers can be effectively obfuscated with the proposed methodology.

The actual integration and management of these obfuscated priorities is not in the scope of this paper. However, since encryption is incrementally introduced in the

automotive domain, handling obfuscated priorities may utilize the same integration and management approaches that are used for authentication and encryption keys.

REFERENCES

AUTOSAR GbR. 2014. Specification of RTE, Version 4.2.1. (2014). http://www.autosar.org/.

L. L. Bello. 2011. The case for ethernet in automotive communications. *ACM SIGBED Review* 8, 4 (2011), 7–15. DOI:http://dx.doi.org/10.1145/2095256.2095257

Bosch. 1991. Controller Area Network, Version 2.0b. (1991). http://www.can.bosch.com/.

P. Caliebe, C. Lauer, and R. German. 2011. Flexible integration testing of automotive ECUs by combining AUTOSAR and XCP. In *Proceedings of International Conference on Computer Applications and Industrial Electronics (ICCAIE 2011)*. 67–72. DOI:http://dx.doi.org/10.1109/ICCAIE.2011.6162106

S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno. 2011. Comprehensive experimental analyses of automotive attack surfaces. In *Proceedings of the 20th USENIX conference on Security (USENIX 2011)*.

A. Davare, Q. Zhu, M. Di Natale, C. Pinello, S. Kanajan, and A. Sangiovanni-Vincentelli. 2007. Period Optimization for Hard Real-time Distributed Automotive Systems. In *Proceedings of the 44th Design Automation Conference (DAC 2007)*. 278–283. DOI:http://dx.doi.org/10.1145/1278480.1278553

R.I. Davis, A. Burns, R.J. Bril, and J.J. Lukkien. 2007. Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised. *Real-Time Systems* 35, 3 (2007), 239–272. DOI:http://dx.doi.org/10.1007/s11241-007-9012-7

M. Di Natale and A. Sangiovanni-Vincentelli. 2010. Moving from federated to integrated architectures in automotive: The role of standards, methods and tools. *Proc. IEEE* 98, 4 (2010), 603–620. DOI:http://dx.doi.org/10.1109/JPROC.2009.2039550

M. Di Natale and H. Zeng. 2010. System identification and extraction of timing properties from controller area network (CAN) message traces. In *Proceedings of the Conference on Emerging Technologies and Factory Automation (ETFA)*. 1–8. DOI:http://dx.doi.org/10.1109/ETFA.2010.5641332

FlexRay Consortium. 2005. FlexRay Communications Systems - Protocol Specification Version 2.1 Rev. A. (2005). http://www.flexray.com.

Gurobi Optimization, Inc. 2015. Gurobi Optimizer Reference Manual. (2015). http://www.gurobi.com

A. Hamann, M. Jersak, K. Richter, and R. Ernst. 2004. Design Space Exploration and System Optimization with SymTA/S – Symbolic Timing Analysis for Systems. In *Proceedings of the 25th IEEE Real-Time Systems Symposium (RTSS 2004)*. 469–478. DOI:http://dx.doi.org/10.1109/REAL.2004.17

G. Han, H. Zeng, Y. Li, and W. Dou. 2014. SAFE: Security-Aware FlexRay Scheduling Engine. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE 2014)*. 1–4. DOI:http://dx.doi.org/10.7873/DATE2014.021

John Harding, Gregory Powell, Rebecca Yoon, Joshua Fikentscher, Charlene Doyle, Dana Sade, Mike Lukuc, Jim Simons, and Jing Wang. 2014. *Vehicle-to-vehicle Communications: Readiness of V2V Technology for Application*. Technical Report DOT HS 812 014.

F. Hartwich. 2012. CAN with flexible data-rate. In *Proceedings of the 13th International CAN Conference (iCC 2012)*. 10–19.

M. Joseph and P. Pandya. 1986. Finding Response Times in a Real-Time System. *Comput. J.* 29, 5 (1986), 390–395. DOI:http://dx.doi.org/10.1093/comjnl/29.5.390

K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage. 2010. Experimental Security Analysis of a Modern Automobile. In *Proceeding of the 2010 IEEE Symposium on Security and Privacy (SP)*. 447–462. DOI:http://dx.doi.org/10.1109/SP.2010.34

J. Lehoczky. 1990. Fixed Priority Scheduling of Periodic Task Sets with Arbitrary Deadlines. In *Proceedings of the 11th International IEEE Real-Time Systems Symposium (RTSS 1990)*. 201–209. DOI:http://dx.doi.org/10.1109/REAL.1990.128748

C.-W. Lin, Q. Zhu, C. Phung, and A. Sangiovanni-Vincentelli. 2013. Security-aware mapping for CAN-based real-time distributed automotive systems. In *Proceedings of the International Conference on Computer-Aided Design (ICCAD 2013)*. 115–121. DOI:http://dx.doi.org/10.1109/ICCAD.2013.6691106

C.-W. Lin, Q. Zhu, and A Sangiovanni-Vincentelli. 2014. Security-Aware Modeling and Efficient Mapping for CAN-Based Real-Time Distributed Automotive Systems. *IEEE Embedded Systems Letters* PP, 99 (2014). DOI:http://dx.doi.org/10.1109/LES.2014.2354011

B. Lisper and P. Mellgren. 2001. Response-time Calculation and Priority Assignment with Integer Programming Methods. In *Proceedings of the Work-in-progress and Industrial Sessions at the 13th Euromicro Conference on Real-Time Systems (ECRTS 2001)*. 13–16.

M. Lukasiewycz, S. Steinhorst, and S. Chakraborty. 2013. Priority Assignment for Event-triggered Systems using Mathematical Programming. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE 2013)*. 982–987. DOI:http://dx.doi.org/10.7873/DATE.2013.205

S. Matic and T. Henzinger. 2005. Trading end-to-end latency for composability. In *Proceedings of the 26th International IEEE Real-Time Systems Symposium (RTSS 2005)*. 99–110. DOI:http://dx.doi.org/10.1109/RTSS.2005.43

A. Metzner and C. Herde. 2006. Rtsat – an optimal and efficient approach to the task allocation problem in distributed architectures. In *Proceedings of the 27th International IEEE Real-Time Systems Symposium (RTSS 2006)*. 147–158. DOI:http://dx.doi.org/10.1109/RTSS.2006.44

C. Miller and C. Valasek. 2013. Adventures in Automotive Networks and Control Units. In *Proceedings of DEF CON*.

C. Miller and C. Valasek. 2014. A Survey of Remote Automotive Attack Surfaces. In *Proceedings of Black Hat*.

P. Mundhenk, S. Steinhorst, M. Lukasiewycz, S. A. Fahmy, and S. Chakraborty. 2015. Lightweight Authentication for Secure Automotive Networks. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE 2015)*.

F. Reimann, M. Glaß, C. Haubelt, M. Eberl, and J. Teich. 2010. Improving Platform-based System Synthesis by Satisfiability Modulo Theories Solving. In *Proceedings of the 8th International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*. 135–144. DOI:http://dx.doi.org/10.1145/1878961.1878986

F. Reimann, M. Lukasiewycz, M. Glass, C. Haubelt, and J. Teich. 2011. Symbolic System Synthesis in the Presence of Stringent Real-time Constraints. In *Proceedings of the 48th Design Automation Conference (DAC 2011)*. 393–398. DOI:http://dx.doi.org/10.1145/2024724.2024817

F. Sagstetter, M. Lukasiewycz, S. Steinhorst, M. Wolf, A. Bouard, W. R. Harris, S. Jha, T. Peyrin, A. Poschmann, and S. Chakraborty. 2013. Security challenges in automotive hardware/software architecture design. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE 2013)*. 458–463. DOI:http://dx.doi.org/10.7873/DATE.2013.102

H. Shacham, M. Page, B. Pfaff, E. Goh, N. Modadugu, and D. Boneh. 2004. On the Effectiveness of Address-space Randomization. In *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS 2004)*. 298–307. DOI:http://dx.doi.org/10.1145/1030083.1030124

Lance Spitzner. 2003. *Honeypots: Tracking Hackers*. Vol. 1. Addison-Wesley Reading.

K. Tindell, A. Burns, and A. Wellings. 1995. Calculating controller area network (CAN) message response times. *Control Engineering Practice* 3, 8 (1995), 1163–1169. DOI:http://dx.doi.org/10.1016/0967-0661(95)00112-8

Ken Tindell and Hans Hansson. 1995. *Real Time Systems by Fixed Priority Scheduling*. Technical Report. Department of Computer Systems - Uppsala University.

B. Wilhelm. 1997. Platform and modular concepts at Volkswagen their effects on the assembly process. In *Transforming Automobile Assembly*. Springer, 146–156. DOI:http://dx.doi.org/10.1007/978-3-642-60374-7_12

S. Woo, H.J. Jo, and D.H. Lee. 2014. A Practical Wireless Attack on the Connected Car and Security Protocol for In-Vehicle CAN. *IEEE Transactions on Intelligent Transportation Systems* PP, 99 (2014), 1–14. DOI:http://dx.doi.org/10.1109/TITS.2014.2351612

R. Zalman and A. Mayer. 2014. A Secure but Still Safe and Low Cost Automotive Communication Technique. In *Proceedings of the 51st Design Automation Conference (DAC 2014)*. 1–5. DOI:http://dx.doi.org/10.1145/2593069.2603850

W. Zheng, M. Di Natale, C. Pinello, P. Giusto, and A. Sangiovanni-Vincentelli. 2007. Synthesis of task and message activation models in real-time distributed automotive systems. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE 2007)*. 93–98. DOI:http://dx.doi.org/10.1109/DATE.2007.364573

Q. Zhu, H. Zeng, W. Zheng, M. Di Natale, and A. Sangiovanni-Vincentelli. 2012. Optimization of task allocation and priority assignment in hard real-time distributed systems. *ACM Transactions on Embedded Computing Systems (TECS)* 11, 4 (2012), 85. DOI:http://dx.doi.org/10.1145/2362336.2362352