A Generic Service-Oriented Function Offloading Framework for Connected Automated Vehicles

Robin Dehler and Michael Buchholz

This paper has been accepted for publication in the journal IEEE Robotics and Automation Letters (RA-L), submitted in January 2025, accepted in March 2025. This is the accepted version of the paper, which has not been fully edited and the layout may differ from the original publication.

Citation information of the original publication:

R. Dehler, and M. Buchholz, "A Generic Service-Oriented Function Offloading Framework for Connected Automated Vehicles," IEEE Robotics and Automation Letters, pp. 1–8, 2025, doi: 10.1109/LRA.2025.3555939.

A Generic Service-Oriented Function Offloading Framework for Connected Automated Vehicles

Robin Dehler[1] and Michael Buchholz[1]

***Abstract*—Function offloading is a promising solution to address limitations concerning computational capacity and available energy of Connected Automated Vehicles (CAVs) or other autonomous robots by distributing computational tasks between local and remote computing devices in form of distributed services. This paper presents a generic function offloading framework that can be used to offload an arbitrary set of computational tasks with a focus on autonomous driving. To provide flexibility, the function offloading framework is designed to incorporate different offloading decision making algorithms and quality of service (QoS) requirements that can be adjusted to different scenarios or the objectives of the CAVs. With a focus on the applicability, we propose an efficient location-based approach, where the decision whether tasks are processed locally or remotely depends on the location of the CAV. We apply the proposed framework on the use case of service-oriented trajectory planning, where we offload the trajectory planning task of CAVs to a Multi-Access Edge Computing (MEC) server. The evaluation is conducted in both simulation and real-world application. It demonstrates the potential of the function offloading framework to guarantee the QoS for trajectory planning while improving the computational efficiency of the CAVs. Moreover, the simulation results also show the adaptability of the framework to diverse scenarios involving simultaneous offloading requests from multiple CAVs.**

***Index Terms*—Intelligent Transportation Systems, Distributed Robot Systems, Networked Robots**

I. Introduction

FOR a reliable and safe application, many functions of autonomous systems require elaborate algorithms with high demands on hardware usage and energy consumption. Due to the often dynamic nature of autonomous systems, e.g., autonomous electric vehicles, both the available hardware and battery are limited. However, the functions of connected automated vehicles (CAVs) are nowadays no longer limited exclusively to hardware and software within the vehicles, but also include both hardware and software components that are employed on external computational units, like servers.

Function offloading is one of the common approaches to use the extended capabilities of such hardware by a CAV. Empowered through the development of wireless communication technologies, e.g., vehicle-to-anything (V2X), CAVs can offload specific functions with the goal to decrease their own overall workload or energy consumption. A wide variety of functions can be considered for function offloading, ranging from infotainment and comfort applications with lower quality of service (QoS) demands, e.g., automatic air conditioning control systems [1], to safety-critical autonomous driving functions with higher QoS demands.

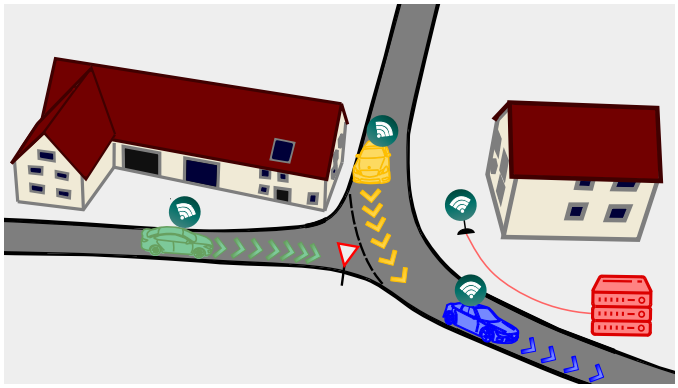


Fig. 1. Considered intersection scenario for service-oriented function offloading including multiple CAVs and a MEC server.

For function offloading, different computing technologies were developed, two prominent ones being Mobile Cloud Computing (MCC) and Multi-Access Edge Computing (MEC). MCC is the technology of using powerful cloud servers as additional hardware. However, MCC has the disadvantage of highly variable latency in the communication with the autonomous system, which limits its application for the often latency-sensitive functions of CAVs [2]. MEC provides an alternative to MCC to overcome its limitations by employing servers at the edge of the network with better performance regarding latency. Consequently, MEC is an important technology for function offloading.

Service-orientation is another trend for autonomous driving and complex systems in general. For service-oriented architectures (SOAs), each function of a CAV can be considered as a service that can easily be modified or exchanged without the need to change the entire computational stack. The concept of SOA is very promising, as it offers the software flexibility needed for autonomous driving [3]. In combination, the two technologies function offloading and SOA provide a generic framework for autonomous driving.

A lot of research on function offloading focuses on algorithms for function offloading decision making and resource allocation, with the objective to decide which functions should be offloaded from autonomous devices to the servers. The

Manuscript received: 10 January 2025; Accepted 11 March 2025.

This paper was recommended for publication by Editor M. Ani Hsieh upon evaluation of the Associate Editor and Reviewers' comments.

This work has been financially supported by the Federal Ministry of Education and Research (project autotech.agil, FKZ 01IS22088W).

[1]The authors are with the Institute of Measurement, Control and Microtechnology, Ulm University, D-89081 Ulm, Germany.

E-mail addresses: {firstname}.{lastname}@uni-ulm.de

Digital Object Identifier (DOI): see top of this page.

function offloading decision making and resource allocation problem is commonly described as an optimization problem considering, e.g., transmit power, bandwidth, computational capacities, energy consumption, and delay requirements [4]. In contrast to the literature, this work aims to provide an overall framework for function offloading that includes the required management functionalities to offload tasks or functions. The function offloading decision making algorithms from the literature can, e.g., be integrated as one part into the proposed framework. Further aspects of function offloading, such as schedulability, security, and fault-tolerance analysis, are outside the scope of this paper and will be considered in future work.

We consider the distribution of a function on different computers as different services that, in the concept of SOA, are easily adaptable and interchangeable by using the framework. The proposed Service-Oriented Function Offloading Framework (SOFOF) comprises both instances of function offloading, i.e., service providers that provide services for others and service requesters that request to offload tasks. In combination, the instances for service providers and requesters are responsible for managing the state transitions of services on multiple devices. Thus, if not used within an existing SOA, the framework can also operate as the service orchestrator within the concept of the automotive service-oriented architecture (ASOA) [3], otherwise, the respective functionalities can also be integrated into the existing orchestrator. In the following, we will concentrate on SOFOF and its special functionalities and refer to, e.g., [3], for more detailed discussions on the ASOA. Using the framework can improve the computation of CAVs by increasing the number of services that can be used for autonomous driving tasks.

For the evaluation, we consider a scenario at a real-world suburban traffic junction with a deployed MEC server, as seen in Fig. 1. We apply and evaluate the framework for the use case of service-oriented trajectory planning in both simulation and real-world application on our test site [5].

Our main contributions are as follows:

- With SOFOF, we present a generic framework that can be used by both function offloading instances, i.e., service provider and service requester.
- We propose an efficient location-based function offloading decision making algorithm, where the decision depends on the location of the CAV and an offloading area dedicated to the server.
- We analyze SOFOF on the application of the use case of service-oriented trajectory planning.
- We extensively evaluate the framework with the use case of service-oriented trajectory planning in both simulation and our real-world autonomous driving test vehicle, showing the applicability of the approach especially for real-world scenarios.

II. Related Work

The research of function offloading mainly addresses the optimization of computational efficiency and resource management in distributed systems. To solve the resulting optimization problems, most approaches propose solutions involving decision making and resource allocation algorithms. The most prominent approaches can be categorized into Mixed-Integer Programming (MIP) approaches, Lyapunov optimization techniques, and approaches using Machine Learning (ML) [6], [7].

When addressed as a MIP, the optimization problem is generally NP-hard, scaling with the number of remote agents, e.g., autonomous vehicles [8]. Thus, the problem is often solved using game theory [9], [10]. Applying game theory, vehicles are players competing for the available resources while considering additional aspects, e.g., a maximum latency. The main advantages of Lyapunov optimization techniques are direct stability analysis and relatively low algorithm complexity [6]. However, choosing an appropriate Lyapunov function is not trivial. Many ML concepts describe the offloading problem as a Markov Decision Process (MDP) and use Reinforcement Learning (RL) for making offloading decisions [11], [12], [13], [14], [15]. Since each concept provides different advantages, they are often combined to benefit from them while reducing disadvantages, e.g., [16], [17], [18]. Note that all of the presented approaches evaluate the proposed offloading algorithm in simulation without real-world analysis.

Another research focus in the area of distributed systems for autonomous driving addresses the usage of different services with a focus on SOAs. It has been established that the concept of SOA is suitable for meeting the requirements of CAVs [19]. A version of a SOA specifically for autonomous driving, called ASOA, was introduced in [3].

Since distributed systems pose greater challenges on safety and security, safety and security analysis is of research interest [20], [19], [21]. In [22], the service of vehicle collaboration is used in combination with a digital twin for collaborative and distributed automated driving. Another approach for cooperative driving in critical sections is proposed in [23]. In [1], the concept of SOA is extended to cloud architectures and specific requirements are analyzed. The authors propose an orchestrator design that manages the execution of both local and cloud services.

Similarly to [1], this work combines function offloading and the concept of SOA in a way that the proposed framework can be employed to use different distributed services for connected automated driving. Compared to the state of the art, we approach the function offloading task in a more general perspective, combining different components, such as function offloading decision making, resource allocation, service orchestration, and QoS considerations into one framework. The service-oriented approach allows flexible interchangeability of different components such that existing algorithms, e.g., for function offloading decision making, can be easily integrated into the framework.

III. Function Offloading Framework

The objective of function offloading is commonly described using an optimization function that consists of different components, e.g., the minimization of latency, the reduction of energy consumption, or the maximization of QoS. Our approach to service-oriented function offloading focuses especially on the QoS and on the improvements that additional

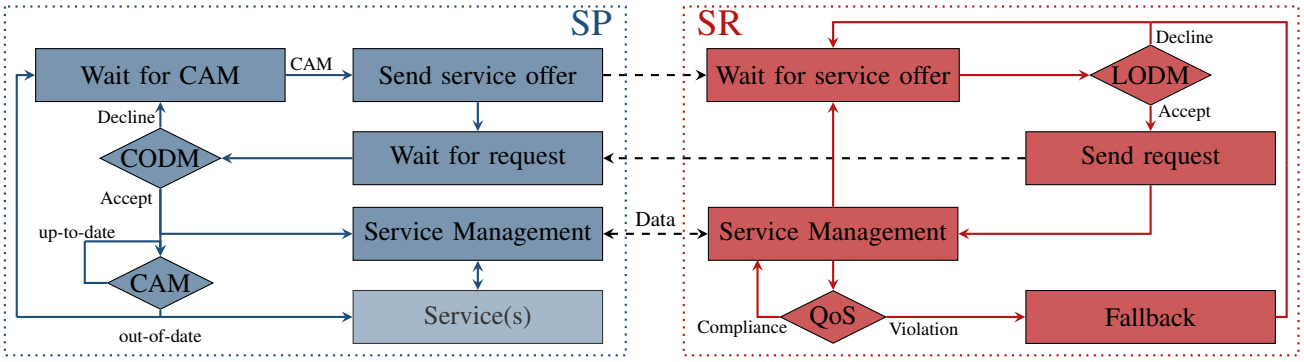


Fig. 2. Overview of SOFOF, simplified for one SP and one SR. After an offloading request is denoted as successful with either the local (LODM) or centralized offloading decision making (CODM), the two service management instances handle the data flow for the offloaded services, respectively. A QoS check on the SR initiates a fallback if the QoS requirements are violated. The CAM check on the SP continuously checks the position of the CAV.

services on distributed hardware can provide compared to only local execution. For that, we consider traffic scenarios in a suburban area with multiple CAVs and a single MEC server unit similar to Fig. 1. The CAVs are considered to continuously send Cooperative Awareness Messages (CAMs) as they are defined by the European Telecommunications Standards Institute (ETSI) [24].

SOFOF comprises two distinct instances: It either serves as a service provider (SP) or a service requester (SR). In its role as a SP, SOFOF handles incoming service requests and activates and manages services for accepted requests. As a SR, its purpose is to analyze which tasks should be offloaded, request the respective services, and manage incoming data. Figure 2 shows the elements of SOFOF for both roles. In the following, the two different operating modes are explained.

A. Service Provider

As shown on the left of Fig. 2, when operating as a SP, the first task is to wait for CAVs to communicate their position to the server, e.g., through CAMs. For clarity of the figure, the separate module of the CAV for the continuous sending of CAM messages is not visualized. When a CAV is detected by a received CAM, the SP signalizes its availability to the SR instance of the CAV in the form of a service offer message and waits for a request of the detected CAV. After receiving a request through a particular request message from the SR, a centralized offloading decision making (CODM) algorithm decides whether the request for the specific service(s) is accepted or rejected. The generalized structure of our framework allows for a flexible interchangeability of the CODM algorithm. If accepted, SOFOF executes each accepted service in a separate thread and manages the services through the service management. The block for the activated service(s) for this SR are visualized in lighter color in Fig. 2, since it is not directly a component of the framework.

For some services, additional data from the served SR is required. The service management accounts for that by transforming and forwarding the received data to the corresponding service in the required format. The data generated by the services is received by the service management unit and forwarded through V2X communication to the SRs.

The described process chain is executed independently for each SR connected to the SP. However, dependent on the implementation, the CODM knows about all other ongoing requests and running offloading services, thus can take the current overall capacity utilization of the SR into account.

B. Service Requester

When acting as a SR, SOFOF first waits for incoming service offers from SP instances. Note that the continuous sending of the CAMs, which are necessary on the SP side, is done in parallel in a separate module, which is not shown to keep the figure clearly arranged. When an incoming service offer is received, a local offloading decision making (LODM) algorithm on the SR side evaluates the benefit of the provided services. Similarly to the CODM, the LODM is easily exchangeable. If the SR wants to use some of the provided services, a request message specifying the requested services is sent to the corresponding SP.

After sending a request, the service management waits for the incoming data for the respective services. After the arrival of the first data for an offloaded function, the function is deactivated and the received data is used for further functions, e.g., vehicle control.

Some services have specific QoS requirements, e.g., maximum latency or minimum frequency of received data. A QoS control instance can be employed to each offloaded service. Thus, different QoS profiles can be adjusted to each service depending on the QoS requirements. If the QoS requirements of a service are violated, a fallback is activated through the service management that stops the offloading of that particular service and simultaneously activates the service on the SR to ensure the QoS for the function, respectively.

IV. Service-Oriented Trajectory Planning

SOFOF serves as a generic framework to handle and manage an arbitrary number of different services for function offloading. In this paper, we focus on one use case in particular, i.e., service-oriented trajectory planning for CAVs. The objective is to use function offloading to provide the service *trajectory planning on the MEC server* (MEC-TPL). The SRs can request and use the service MEC-TPL as an addition to

the service *local trajectory planning* (CAV-TPL). Within this paper, we integrate our framework into our existing, traditional architecture for autonomous driving. We only consider the function trajectory planning as a service, all other parts are classical functions onboard the vehicle. Within the concept of ASOA, our framework hence operates as the orchestrator for multiple trajectory planning services.

For our use case, we developed two algorithms for CODM and LODM that are explained in the following.

A. Centralized Offloading Decision Making

A CODM employed on the SP is used to accept or reject incoming service requests. In this work, we use a location-based CODM algorithm that depends on two conditions. The SP has stored maps and an offloading area in which function offloading is possible in the form of polygon points. The offloading area can, e.g., be determined dependent on the network connection. Within the offloading request, the SRs send the planned route to the SP. Initially, the received route is compared with the maps available to the SP. If a corresponding map is present, a second condition checks whether the route is long enough in the area where offloading is possible using the ray-casting algorithm [25]. For the distance, the received route waypoints are analyzed until the first waypoint is outside the polygon. Since the route information does not include vehicle speeds, a constant velocity (CV) model is used on the current speed of the SR. The CV model serves as a simple analysis instead of a forward planning approach, e.g., considering road topology and velocity limits, based on the trade-off between computational complexity and performance of the CODM algorithm. The time is taken from the sum of the Euclidean distances between successive waypoints until the last waypoint in the polygon area given the constant velocity v_c, see Eq. (1).

For the condition, a threshold value t_{min} is used to decide whether offloading is worthwhile. Thus, the condition can be stated as

$$t_{min} \leq \sum_{i=1}^{l} \frac{\sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}}{v_c}, \qquad (1)$$

with x_i and y_i being the x- and y-coordinates of waypoint i and l the index of the last waypoint in the polygon area.

While the service management handles the activated services, the latest CAM of the served CAV is constantly updated. If the CAV leaves the offloading area or if the latest CAM is older than two seconds, the offloading is terminated and the activated services are deactivated.

B. Local Offloading Decision Making

A received service offer contains information about the position of the SP pos_{SP}. A LODM algorithm is used to accept or decline the offloading offer. We again use a location-based decision making process that is summarized in Alg. 1. For the LODM, an arbitrary offloading radius r_{off} can be set. This radius determines the maximum distance that a SR must have to the respective connection point to the SP so that function offloading is possible.

Algorithm 1 Local Offloading Decision Making

Input: path_{CAV}, SR position pos_{last}, r_{off}, pos_{SP}, d_{min}
Output: Bool value for decision

1: $d_{\text{passed}} = 0$, $\text{pos}_{\text{curr}} = \text{path}_{\text{CAV}}[1]$
2: **while** Distance of pos_{curr} to pos_{SP} < r_{off} **do**
3: $\quad d_{\text{pos}} =$ Distance between pos_{curr} and pos_{last}
4: $\quad d_{\text{passed}} \leftarrow d_{\text{passed}} + d_{\text{pos}}$
5: $\quad$ **if** $d_{\text{passed}} > d_{\text{min}}$ **then**
6: $\quad\quad$ **return** True
7: $\quad \text{pos}_{\text{last}} \leftarrow \text{pos}_{\text{curr}}$, $\text{pos}_{\text{curr}} \leftarrow$ next point of path_{CAV}
8: **return** False

Alg. 1 iterates through the planned path of the CAV path_{CAV} and compares the distance of the points to the SP connection point pos_{SP} with r_{off} (line 2). As long as the path is within r_{off}, the distance d_{pos} from the current position pos_{curr} to the previous pos_{pos} is added to d_{passed} (lines 3 - 4). As soon as d_{passed} is greater than a minimum distance d_{min}, which means that path_{CAV} is long enough within r_{off} to pos_{SP}, the function offloading request is accepted (lines 5 - 6).

C. Trajectory Planning

We use common ETSI Intelligent Transportation Systems (ITS) messages [26] for sending the data from the CAV to the MEC server and vice versa via a cellular network. The particular trajectory planning algorithm for both the CAV-TPL and the MEC-TPL was introduced in [27]. It uses, in a nutshell, an Augmented Lagrangian-Iterative Linear Quadratic Regulator (AL-ILQR) optimization method in a Model Predictive Control (MPC) fashion to compute trajectories. Note that the trajectory planning algorithms for different services can also be different, e.g., using a more complex planner for the MEC-TPL to make use of the powerful hardware on the MEC server for calculating higher quality trajectories as an additional benefit for function offloading.

As mentioned before, the service management instances of SOFOF need to manage the data flow for the corresponding services. The trajectory planning framework takes as input an environment model and the ego motion of the CAV. The information about the ego motion can be created using the CAM. The environment model of the CAV is sent to the MEC server using the ETSI ITS Collective Perception Message (CPM) [28]. Another benefit of trajectory planning on the MEC server is that the server can directly benefit from the extended environment model through infrastructure sensors. The trajectories calculated by the MEC-TPL are received by the SP instance and then encoded into a custom Maneuver Coordination Message (MCM) that was proposed in [29]. We make use of our custom MCM, because the MCM is currently not yet standardized. Specifically, the trajectory is encoded into the *SuggestedManeuverContainer* of the MCM from [29] and sent to the CAVs. The service management instance on the CAV receives the MCMs, extracts the trajectories and then forwards them to the vehicle control system. The particular data flow of the ETSI ITS messages during activated function

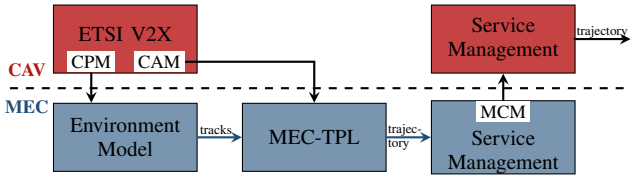


Fig. 3. Data flow between the different components of the function offloading use case service-oriented trajectory planning.

offloading for service-oriented trajectory planning is visualized in Fig. 3.

D. Quality of Service Requirements

Trajectory planning is one of the last steps in the execution chain of vehicle automation. The subsequent control system actuates the vehicle control, such as adjusting the steering angle or acceleration. Consequently, trajectory planning has strict QoS requirements. Thus, for the service-oriented trajectory planning, we use a maximum latency $l_{\max}$ and a maximum inter-arrival time $\Delta t_{\max}$ as QoS requirements. The latency is the difference of the trajectory generation time t_{creation} to the time when the trajectory is received $t_{\text{receiving}}$. The inter-arrival time is defined as the time between two consecutively received trajectories t_i and t_{i+1}, $i \in \mathbb{N}_0$. If the two requirements summarized with

$$\begin{aligned} l_{\max} &\leq t_{\text{receiving}} - t_{\text{creation}} \\ \Delta t_{\max} &\leq t_{i+1} - t_i \end{aligned} \tag{2}$$

are not fulfilled, the CAV-TPL is re-activated and the SR communicates the termination of the service to the SP.

V. Evaluation

We evaluate SOFOF both in simulation and in a real-world scenario with the considered use case of service-oriented trajectory planning. SOFOF is implemented as a ROS2 node in our autonomous driving stack consisting of multiple nodes for perception and planning tasks and a V2X module. The dynamic node handling is achieved through the concept of ROS2 lifecycle nodes.

A. Evaluation in Simulation

In the simulation, we compare the application of SOFOF for service-oriented trajectory planning.

1) Simulation Setup: We use a ROS2-based software-in-the-loop (SIL) simulation framework that was proposed in [30]. The SIL simulation runs on Ubuntu 22.04 on an AMD Ryzen 9 7950x CPU with 64 GBs RAM. Each simulated CAV allocates its designated ROS2 nodes of the autonomous driving stack including the SOFOF node. The nodes for the simulated MEC server run on separate hardware with an AMD Ryzen 9 3950X CPU with 64 GBs RAM. The two machines are in two distinct local networks that are bridged through a fiber connection. The multi-hop connection aims to better represent the network conditions in a wireless connection. The sending of offloading messages and V2X communication are done

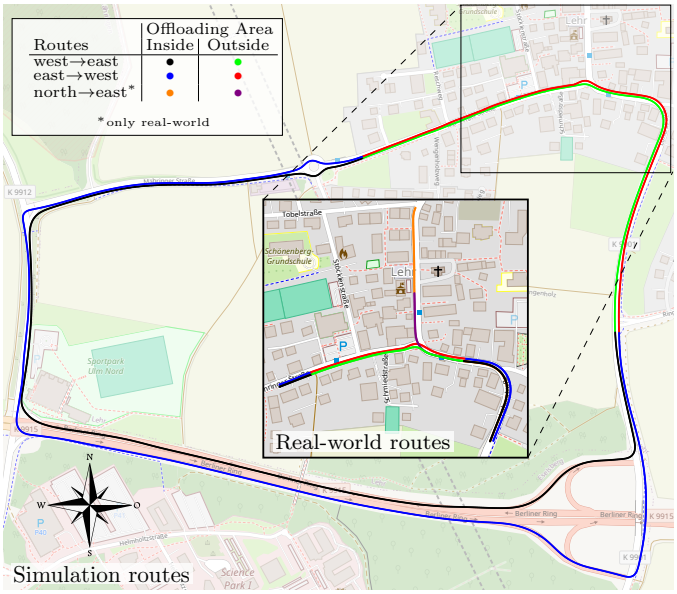


Fig. 4. Real-world street layout with the pilot junction and the two round-trips that were emulated in the simulation in the large picture. The zoomed detailed picture shows the same junction with the routes of the real-world tests. Both pictures use the same color coding for the routes given in the legend. Map from [32].

using an AMQP broker [31] on the machine of the simulated MEC server and different clients for each simulated CAV and the MEC server.

The simulation results involve multiple scenarios with varying number of CAVs ranging from one to six. The street topology of the simulation is equivalent to a real-world street layout in Ulm-Lehr, Germany, with two round-trips of $\sim 2.9\,\text{km}$ each. The equivalent street layout contains a pilot test site that was outlined in [5]. The street layout for both the simulation and the real-world experiments is shown in Fig. 4, where the background shows the round-trips of the simulation and the zoomed traffic junction in the middle the routes for the real-world analysis. The color change shows the different offloading areas, respectively. It can be seen that the areas for the simulations are comparatively larger.

We evaluate different scenarios, where the spawn position and route of the involved CAVs on the round-trips are changed to account for different timing conditions of the requests. For the CODM, we choose the offloading area in a way that $\sim 700\,\text{m}$ of each round-trip is included in the offloading area on the simulated MEC server and an offloading radius of $r_{\text{off}} = 300\,\text{m}$ for the LODM of the CAVs. In the simulations, the CAVs drive the round-trip multiple times. The size of the offloading area results in $\sim \frac{1}{4}$ of the overall length of the round-trip.

2) QoS Requirements: To determine suitable parameters for the QoS requirements, we first analyze the influence of different parameters on different scenarios.

Fig. 5 shows the offloading duration t_{off} of different scenarios involving a different number of vehicles depending on the maximum allowed inter-arrival time $\Delta t_{\max}$. t_{off} is the overall time of how long the trajectory planning service is offloaded

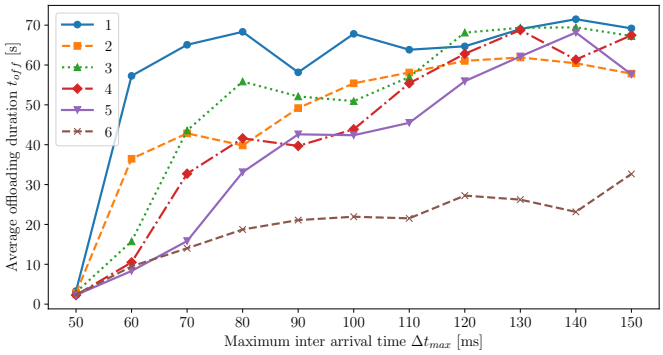


Fig. 5. Average offloading duration t_{off} depending on the maximum inter-arrival time Δt_{max} with scenarios of varying number of CAVs.

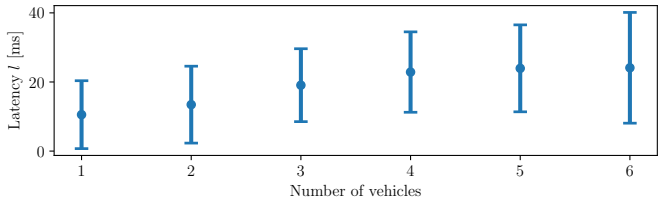


Fig. 6. Mean latency values including the standard deviation depending on the number of simulated vehicles.

for each accepted service request. We ran each scenario with varying CAV count $\in [1, \ldots, 6]$ and $\Delta t_{\text{max}} \in [10, \ldots, 150]$ms for 30 minutes, thus conducting a total of 33 hours of simulation, including numerous round-trips for each simulated CAV. With higher Δt_{max}, more trajectories can be received within the time limit, which means that offloading is possible for a longer time. This can also be seen in Fig. 5, where the offloading duration t_{off} generally increases with higher Δt_{max}. The maximum offloading duration of the scenario is bounded to $\sim 70\,\text{s}$, which corresponds to the time the CAVs spend in the offloading area.

Fig. 5 also shows a slower increase of t_{off}, and thus a generally shorter offloading duration for more vehicles. Especially for scenarios with six CAVs, the maximum offloading duration is lower than for less complex scenarios. The first reason for this is the higher network congestion and workload on the MEC server with more offloading requests resulting in more violations of the QoS requirements, and thus an earlier termination of the offloading. The second, however, results from the simulation setup. With six vehicles present, the machine running the simulation is close to its computational limits, which also reduces the offloading duration for these scenarios. Thus, the network congestion is not solely responsible for the worse performance and a more profound simulation setup might improve the results.

For the analysis shown in Fig. 5, the second parameter of the QoS requirements, i.e., the maximum latency, is set to $l_{\text{max}} = 50\,\text{ms}$. To assess the validity of this value, we stored the latencies of all received trajectories of the simulations. Fig. 6 shows the mean delay values and its standard deviations depending on the number of simulated vehicles in the scenarios. It can be seen that the both the mean latency and the standard deviation increase with higher number of vehicles. This is a result of the higher and more varying network congestion following from more vehicles, respectively. The distribution also shows that, with the given setup, it is highly probable that the latency of a received trajectory is less than $50\,\text{ms}$.

TABLE I
AVERAGE CPU USAGE OF RELEVANT SYSTEMS W.R.T ONE CPU CORE.

System	Simulation	Real-World
TPLa c_{TPLa}	19.5%	116.1%
TPLd c_{TPLd}	8.5%	28.3%
SOFOF SR $c_{\text{SOFOF SR}}$	0.96%	2.61%

Considering the real-time requirements for vehicle control and from the results in Fig. 5, we determine as suitable QoS requirements:

$$\begin{aligned} \Delta t_{\text{max}} &= 100\,\text{ms} \\ l_{\text{max}} &= 50\,\text{ms} \end{aligned} \tag{3}$$

3) CPU Usage: One of the main motivations of function offloading was to reduce the energy consumption of the battery-powered CAVs. For that, we evaluate the average CPU usage in percent in relation to the capacity of one core of the CPU of the corresponding systems with function offloading activated and deactivated, since needed computational power correlates with the energy consumption. It is worth mentioning that the overall CPU usage of all CAVs and the MEC server is evidently higher due to the additional functionality. However, the main focus is the minimization of the hardware consumption of the CAVs where we assume that we have limited power available.

On the CAV, the overall CPU usage without and with SOFOF are characterized by

$$\begin{aligned} c_{\text{without}} &= t_{\text{total}} \cdot c_{\text{TPLa}} \quad \text{and} \\ c_{\text{with}} &= t_{\text{total}} \cdot c_{\text{SOFOF SR}} + t_a c_{\text{TPLa}} + t_d \cdot c_{\text{TPLd}}, \end{aligned} \tag{4}$$

respectively, with t_{total} the total time and t_d, t_a the times with TPL deactivated and activated, respectively. c_{TPLa}, c_{TPLd}, and $c_{\text{SOFOF SR}}$ are the respective average CPU percentages for TPLa, TPLd, and SOFOF as a SR. In another simulation, where we set the QoS requirements as in Eq. (3), we recorded the average CPU usage values, which can be seen in the second column of Table I.

With the main goal being the reduction of CPU usage through function offloading, the objective can be formulated as

$$c_{\text{with}} < c_{\text{without}}. \tag{5}$$

When inserting Eq. (4) into Eq. (5) and with $t_a = t_{\text{tot}} - t_d$, the proportion of how long offloading should be activated w.r.t the total application time to reach the objective is

$$\frac{t_d}{t_{\text{tot}}} > \frac{c_{\text{SOFOF}}}{c_{\text{TPLa}} - c_{\text{TPLd}}} = 0.0793 \;\widehat{=}\; 7.93\,\%, \tag{6}$$

with 0.0793 resulting from the simulated values in Table I.

In our experiments, we also analyzed the CPU usage of the V2X system that is used to send the CAMs, CPMs, and MCMs

from the CAV to the MEC server and vice versa. However, when assuming that CAMs and CPMs are sent even without function offloading for other V2X functionality, the additional usage due to function offloading itself is negligible and thus excluded from the evaluation.

It can be concluded that in the simulation setup, SOFOF can improve the overall CPU workload of the CAVs for service-oriented trajectory planning, if the trajectory planning task is offloaded for at least 7.93%, with great improvements for higher offloading ratios.

B. Real-World Evaluation

Since the latency considerations in the simulation do not necessarily reflect the application of the framework in a real-world scenario, we also deploy SOFOF on our autonomous test vehicle and on the MEC server of our test site [5]. We evaluate the adherence of the QoS requirements in a similar way as in the simulation, showing the robustness towards network effects resulting from the dynamic nature of real-word traffic scenarios.

1) Real-World Setup: We again use ROS2 on the test vehicle and the MEC server, respectively. The autonomous test vehicle is a Mercedes-Benz E-class equipped with an AMD Ryzen Threadripper 3990X CPU with $128\,\mathrm{GBs}$ RAM, while the MEC server comprises an AMD Ryzen Threadripper PRO 5965WX CPU with $178\,\mathrm{GBs}$ RAM. The network connection is established through a 5G network with an AMQP connection to handle the V2X communication. Derived from the simulation, we set the QoS requirements to $\Delta t_{\text{max}} = 100\,\mathrm{ms}$ and $l_{\text{max}} = 50\,\mathrm{ms}$.

For safety reasons, we defined a different offloading area for the CODM on the MEC server compared to the simulations. In particular, the chosen offloading area includes $\sim 220\,\mathrm{m}$ of the predefined routes in the vicinity of our pilot traffic junction, see Fig. 4. The actual possible area could be much larger. We conducted 15 test drives on three different routes through the area shown in Fig.4, where the SOFOF algorithm dynamically changes the trajectory planning service from CAV-TPL to MEC-TPL depending on the position of our autonomous test vehicle and the defined offloading area. Similarly to the simulation analysis, we monitored the needed CPU usage of the relevant ROS2 nodes, i.e., trajectory planning (TPLa and TPLd) and function offloading (SOFOF SR), while we excluded SOFOF SP since it runs on the MEC server with a wired energy supply.

2) CPU Usage: The third column of Table I shows the average observed CPU usage of the different components of the autonomous test vehicle. When comparing the two columns for the simulation and real-world analysis, it is noticeable that the CPU usage of TPLa is significantly higher in the real-world evaluation than in the simulation for the one vehicle. This is a result of a different parameter setup of the TPLa, where the update frequency of the MPC planning algorithm, see [27], is set to $100\,\mathrm{Hz}$ for the real-world application instead of $20\,\mathrm{Hz}$ in the simulation. The higher update frequency ensures safer trajectory planning, as it can adapt faster to changes in the environment of the vehicle. It is worth noting that the QoS requirements are not adjusted, since the chosen values are still valid as a fallback solution if the latency and inter-arrival time are not met. Also, since we take the percentage values in relation to one CPU core, the value can be $> 100\%$, meaning that the process needs multiple CPU cores to run.

The implementation of SOFOF on the real-world test vehicle also needs a higher percentage of the CPU than in the simulation. The reason for this is similar to the higher value of TPLa. Since the published trajectories need to be handled by SOFOF, the CPU usage is consequently higher for a higher publish frequency.

With the objective from Eq. (5), the time ratio for how long the trajectory planning needs to be deactivated in order for the system with SOFOF to use less of the CPU is

$$\frac{t_d}{t_{\text{tot}}} > 0.0615 \;\;\widehat{=}\;\; 6.15\,\%. \tag{7}$$

We also recorded the latency for each trajectory that was sent from the MEC server to our test vehicle. For the real-world scenarios, the mean and standard deviation values were

$$\begin{aligned} l_{\text{avg}}^{\text{real}} &= 11.94\,\mathrm{ms} \\ l_{\text{std}}^{\text{real}} &= 2.98\,\mathrm{ms}. \end{aligned} \tag{8}$$

Especially the value for the standard deviation from Eq. (8) varies with the values for one simulated scenario shown in Fig. 6. In particular, for one vehicle, Fig. 6 shows $l_{\text{avg}}^{\text{sim,1}} = 10.54\,\mathrm{ms}$ and $l_{\text{std}}^{\text{sim,1}} = 9.83\,\mathrm{ms}$. One reason for the lower standard deviation is that in the real-world analysis, only the data relevant for function offloading is sent to the server, while in the simulation additional simulation data needs to be sent to the machine simulating the server.

With the observed values $l_{\text{avg}}^{\text{real}}$, $l_{\text{std}}^{\text{real}}$ and $l_{\text{max}} = 50\,\mathrm{ms}$, a very high percentage of trajectories is received within the maximum delay. Nevertheless, in 2 of the in total 15 conducted scenarios, the function offloading had to be terminated early due to a violation of l_{max} within the offloading area shown in Fig. 4. However, the dynamic switch back to the CAV-TPL resulting from the violation was not noticeable in the driving experience of the passengers, demonstrating the smooth and safe transitions between the trajectory planning services even under varying network conditions.

Therefore, it can be stated that real-world experiments not only support the simulation results but also show that the application of SOFOF can efficiently extend the local computational hardware and software of our autonomous test vehicle.

VI. Conclusion and Future Work

This paper presented SOFOF, a generic function offloading framework in the context of SOAs that can extend the number of available services for the task of autonomous driving. Exemplarily, the task of offloading the trajectory planning was introduced as a use case and was handled by the framework. In the evaluation, the benefit of SOFOF in improving hardware usage on the CAVs was shown in simulation. The real-world evaluation showed the applicability of the framework to actual scenarios, showing that function offloading holds great promise for the development of future mobility systems.

Due to the generic design, SOFOF opens a wide range of further research. Our future work will focus on enhancing the system to further benefit from function offloading by involving multiple different services including cooperative functions, e.g., cooperative intersection management [33]. We will also focus on enhancing the simulation setup to better evaluate the scalability of SOFOF and consider the aspect of safety by not only using the QoS requirements, but also analyzing the quality of the offloaded tasks through analyzing the downstream data of the different services.

REFERENCES

[1] M. Sommer, H. Guissouma, M. Schindewolf, and E. Sax, "An orchestrator for the dynamic extension of automotive e/e architectures to the cloud," in *Service-Oriented Computing*, M. Aiello, J. Barzen, S. Dustdar, and F. Leymann, Eds. Cham: Springer Nature Switzerland, 2025, pp. 24–41.

[2] H. Wang, T. Liu, B. Kim, C.-W. Lin, S. Shiraishi, J. Xie, and Z. Han, "Architectural Design Alternatives Based on Cloud/Edge/Fog Computing for Connected Vehicles," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2349–2377, 2020.

[3] A. Kampmann, B. Alrifaee, M. Kohout, A. Wüstenberg, T. Woopen, M. Nolte, L. Eckstein, and S. Kowalewski, "A Dynamic Service-Oriented Software Architecture for Highly Automated Vehicles," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 2101–2108.

[4] J. Du, L. Zhao, J. Feng, and X. Chu, "Computation Offloading and Resource Allocation in Mixed Fog/Cloud Computing Systems with Min-Max Fairness Guarantee," *IEEE Transactions on Communications*, vol. 66, no. 4, pp. 1594–1608, 2018.

[5] M. Buchholz, J. Müller, M. Herrmann, J. Strohbeck, B. Völz, M. Maier, J. Paczia, O. Stein, H. Rehborn, and R.-W. Henn, "Handling Occlusions in Automated Driving Using a Multiaccess Edge Computing Server-Based Environment Model From Infrastructure Sensors," *IEEE Intelligent Transportation Systems Magazine*, vol. 14, no. 3, pp. 106–120, 2022.

[6] F. Saeik, M. Avgeris, D. Spatharakis, N. Santi, D. Dechouniotis, J. Violos, A. Leivadeas, N. Athanasopoulos, N. Mitton, and S. Papavassiliou, "Task offloading in Edge and Cloud Computing: A survey on mathematical, artificial intelligence and control theory solutions," *Computer Networks*, vol. 195, p. 108177, 2021.

[7] A. Acheampong, Y. Zhang, X. Xu, and D. Kumah, "A Review of the Current Task Offloading Algorithms, Strategies and Approach in Edge Computing Systems," *Computer Modeling in Engineering & Sciences*, vol. 134, pp. 1–54, 08 2022.

[8] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-Efficient Offloading for Mobile Edge Computing in 5G Heterogeneous Networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016.

[9] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation Offloading and Resource Allocation For Cloud Assisted Mobile Edge Computing in Vehicular Networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7944–7956, 2019.

[10] Z. Ning, P. Dong, X. Wang, X. Hu, J. Liu, L. Guo, B. Hu, R. Y. K. Kwok, and V. C. M. Leung, "Partial Computation Offloading and Adaptive Task Scheduling for 5G-Enabled Vehicular Networks," *IEEE Transactions on Mobile Computing*, vol. 21, no. 4, pp. 1319–1333, 2022.

[11] N. Zhao, Z. Ye, Y. Pei, Y.-C. Liang, and D. Niyato, "Multi-Agent Deep Reinforcement Learning for Task Offloading in UAV-Assisted Mobile Edge Computing," *IEEE Transactions on Wireless Communications*, vol. 21, no. 9, pp. 6949–6960, 2022.

[12] J. Wang, J. Hu, G. Min, A. Y. Zomaya, and N. Georgalas, "Fast Adaptive Task Offloading in Edge Computing Based on Meta Reinforcement Learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 242–253, 2021.

[13] Q. Luo, T. H. Luan, W. Shi, and P. Fan, "Deep Reinforcement Learning Based Computation Offloading and Trajectory Planning for Multi-UAV Cooperative Target Search," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 2, pp. 504–520, 2023.

[14] H. Li, K. D. R. Assis, S. Yan, and D. Simeonidou, "Drl-Based Long-Term Resource Planning for Task Offloading Policies in Multiserver Edge Computing Networks," *IEEE Transactions on Network and Service Management*, vol. 19, no. 4, pp. 4151–4164, 2022.

[15] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep Reinforcement Learning for Online Computation Offloading in Wireless Powered Mobile-Edge Computing Networks," *IEEE Transactions on Mobile Computing*, vol. 19, no. 11, pp. 2581–2593, 2020.

[16] W. Sun, H. Zhang, R. Wang, and Y. Zhang, "Reducing Offloading Latency for Digital Twin Edge Networks in 6G," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 10, pp. 12 240–12 251, 2020.

[17] S. Bi, L. Huang, H. Wang, and Y.-J. A. Zhang, "Lyapunov-Guided Deep Reinforcement Learning for Stable Online Computation Offloading in Mobile-Edge Computing Networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 11, pp. 7519–7537, 2021.

[18] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang, "Deep Reinforcement Learning for Stochastic Computation Offloading in Digital Twin Networks," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 4968–4977, 2021.

[19] S. Kugele, P. Obergfell, M. Broy, O. Creighton, M. Traub, and W. Hopfensitz, "On Service-Orientation for Automotive Software," in *2017 IEEE International Conference on Software Architecture (ICSA)*, 2017, pp. 193–202.

[20] M. Rumez, D. Grimm, R. Kriesten, and E. Sax, "An Overview of Automotive Service-Oriented Architectures and Implications for Security Countermeasures," *IEEE Access*, vol. 8, pp. 221 852–221 870, 2020.

[21] H. Zhu, R. Lu, X. Shen, and X. Lin, "Security in service-oriented vehicular networks," *IEEE Wireless Communications*, vol. 16, no. 4, pp. 16–22, 2009.

[22] Y. Hui, X. Ma, Z. Su, N. Cheng, Z. Yin, T. H. Luan, and Y. Chen, "Collaboration as a Service: Digital-Twin-Enabled Collaborative and Distributed Autonomous Driving," *IEEE Internet of Things Journal*, vol. 9, no. 19, pp. 18 607–18 619, 2022.

[23] K. Zhang, A. Yang, H. Su, A. de La Fortelle, K. Miao, and Y. Yao, "Service-Oriented Cooperation Models and Mechanisms for Heterogeneous Driverless Vehicles at Continuous Static Critical Sections," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 7, pp. 1867–1881, 2017.

[24] ETSI EN 302 637-2, "Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service," Sep. 2014, final draft V1.3.1.

[25] S. D. Roth, "Ray casting for modeling solids," *Computer Graphics and Image Processing*, vol. 18, no. 2, pp. 109–144, 1982.

[26] ETSI ES 202 663, "Intelligent Transport Systems (ITS); European profile standard for the physical and medium access control layer of Intelligent Transport Systems operating in the 5 GHz frequency band," Nov. 2009, final draft V1.1.0.

[27] J. Ruof, M. B. Mertens, M. Buchholz, and K. Dietmayer, "Real-Time Spatial Trajectory Planning for Urban Environments Using Dynamic Optimization," in *2023 IEEE Intelligent Vehicles Symposium (IV)*, 2023, pp. 1–7.

[28] ETSI TR 103 562, "Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Analysis of the Collective Perception Service (CPS); Release 2," Dec. 2019, v2.1.1.

[29] M. B. Mertens, J. Müller, R. Dehler, M. Klimke, M. Maier, S. Gherekhloo, B. Völz, R.-W. Henn, and M. Buchholz, "An Extended Maneuver Coordination Protocol with Support for Urban Scenarios and Mixed Traffic," in *2021 IEEE Vehicular Networking Conference (VNC)*, 2021, pp. 32–35.

[30] J. Strohbeck, J. Müller, A. Holzbock, and M. Buchholz, "DeepSIL: A Software-in-the-Loop Framework for Evaluating Motion Planning Schemes Using Multiple Trajectory Prediction Networks," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 7075–7081.

[31] OASIS, *OASIS Advanced Message Queuing Protocol (AMQP) Version 1.0 Part 3: Messaging*, October 29 2012, OASIS Standard. [Online]. Available: http://docs.oasis-open.org/amqp/core/v1.0/

[32] OpenStreetMap contributors, "Planet dump from https://planet.osm.org ," https://www.openstreetmap.org/#map=16/48.42884/9.96415, 2025.

[33] M. B. Mertens, J. Müller, and M. Buchholz, "Cooperative maneuver planning for mixed traffic at unsignalized intersections using probabilistic predictions," in *2022 IEEE Intelligent Vehicles Symposium (IV)*, 2022, pp. 1174–1180.