Enhancing Real-Time Performance in Resource-Constrained Autonomous Racing Vehicle through Task Offloading

Pascal Reich, Hyunjong Choi
San Diego State University
{preich5404, hyunjong.choi}@sdsu.edu

***Abstract*—Autonomous vehicles must execute critical tasks such as perception and localization with low latency to maintain safety, accuracy, and stability. In autonomous racing, these timing requirements become even more demanding, as vehicles must navigate complex tracks and dynamic obstacles at high speed. On scaled platforms such as F1Tenth, limited onboard computational resources often force trade-offs between localization accuracy and system responsiveness. In this work, we show that high end-to-end latency in the localization pipeline directly affects waypoint selection and increases cross-track error, leading to degraded racing performance. To address this issue, we propose a task offloading architecture that transfers computation-intensive components to an external host system. Experimental results on a physical F1Tenth platform demonstrate that the proposed approach significantly reduces localization latency and onboard CPU utilization, resulting in improved tracking accuracy and more consistent racing performance.**

I. Introduction

Emerging autonomous robotic systems increasingly rely on multi-stage processing pipelines that integrate sensing, state estimation, perception, planning, and control under strict timing constraints. In such systems, end-to-end latency is not merely a performance metric but a correctness condition: delayed state propagation can directly degrade closed-loop control stability and safety. Unlike full-scale autonomous vehicles equipped with high-performance onboard computing platforms, scaled racing systems such as F1Tenth [1] operate on resource-constrained embedded hardware. At the same time, they travel at relatively high speeds on compact tracks, which significantly reduces tolerance to perception and localization delays. The combination of tight timing requirements and limited computational capacity makes maintaining both estimation accuracy and real-time responsiveness challenging.

Traditional navigation approaches, such as gap-following, enable basic obstacle avoidance but lack state awareness and trajectory optimization. Modern racing stacks employ localization and perception pipelines to support map-based navigation and trajectory tracking. However, algorithms such as Adaptive Monte Carlo Localization (AMCL) involve computationally intensive particle updates and scan matching operations, introducing substantial execution variability. As vehicle speed increases, the tolerance to localization latency decreases, creating a tight coupling between pipeline timing and control accuracy.

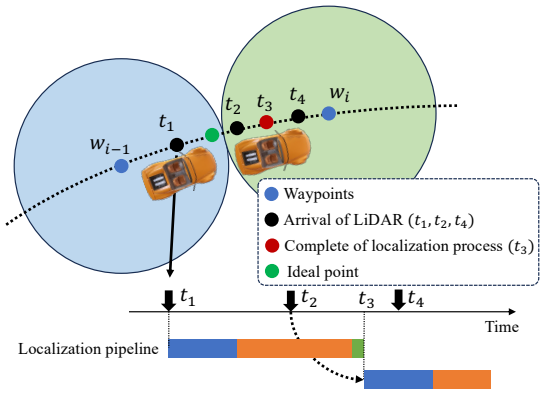


Fig. 1: Waypoint selection error induced by localization latency.

Challenges. Our experimental analysis reveals the following two latency-induced phenomena that significantly impact control quality.

(1) Scan loss due to sequential pipeline execution. AMCL processes LiDAR scans sequentially and employs mutex-protected callbacks to prevent concurrent execution. When the worst-case execution time approaches the LiDAR sampling period, newly arrived scan instances experience phasing delay. Accumulated delay may exceed one sensor period, forcing the system to discard pending scans. This effectively reduces the update rate of the localization pipeline and introduces discontinuities in state estimation, despite moderate average CPU utilization.

(2) Control degradation due to delayed state estimation. Control decisions depend on selecting the nearest waypoint based on the latest pose estimate. As illustrated in Fig. 1, localization delay causes the estimated pose to lag behind the vehicle's true pose. At racing speeds, even millisecond-scale delays correspond to non-negligible spatial displacement (e.g., $0.075\,\mathrm{m}$ for a $0.025\,\mathrm{s}$ delay at $3\,\mathrm{m/s}$). This temporal lag induces incorrect waypoint selection and increases cross-track error, establishing a direct latency-to-control degradation mechanism.

To mitigate latency-induced degradation, we investigate task offloading as a mechanism to reduce the end-to-end latency of computation-intensive pipelines. We design a client-host archi-

tecture that offloads localization computation to an external processing unit while maintaining onboard control execution. The design aims to reduce worst-case latency and stabilize pipeline timing without modifying the control logic.

Contributions. This paper makes the following contributions:

- We experimentally characterize the coupling between end-to-end localization latency and closed-loop tracking error on a physical autonomous racing platform.
- We identify and analyze latency-induced scan loss and waypoint mis-selection as pipeline-level timing effects that are not captured by conventional task-level analysis.
- We design and implement a task offloading architecture that reduces end-to-end latency and improves control stability in resource-constrained autonomous systems.

The remainder of the paper is organized as follows. Sec. II reviews related work. Sec. III describes the system architecture and performance metrics. Sec. IV details the offloading design. Sec. V demonstrates experimental results, and Sec. VI concludes the paper.

II. Related Work

Extensive studies have focused on enhancing localization accuracy and the overall performance of autonomous robotic systems. Gavioli et al. [6] introduce an adaptive localization method that employs a particle filter to improve computational efficiency and localization accuracy under resource-constrained environments. Further algorithmic improvements for localization are explored in [4], where authors optimize the process by selectively focusing on significant particles within a local neighborhood or by employing feature extraction techniques to improve both speed and accuracy. Hardware acceleration has also been applied to localization tasks in Bernardi et al. [3] and others [8] by leveraging FPGA-based implementations to reduce the onboard computational load and enhance vehicle performance in racing contexts. However, studies such as [3, 6, 8] have primarily been conducted in simulation, and other algorithmic approaches [4] have not been evaluated for their practicality in real-world racing applications. Choi et al. [5] introduce a priority-driven chain-aware scheduler to optimize task scheduling while considering chain configurations, however, it does not reduce the processing time for pipelines that are composed of sequentially executed tasks.

Studies on offloading techniques underscore its potential to enhance system performance, especially in resource-constrained environments and edge-computing contexts. Ren et al. [10] and Shu et al. [13] demonstrate latency reduction in video compression and multi-user offloading by leveraging edge clouds in simulation. Similarly, authors in [14] show the effectiveness of offloading through Mobile Edge Computing (MEC) and peer cloud networks in alleviating processing demands on vehicles, although these approaches remain simulation-based. Sarker et al. [12] apply offloading to low-power indoor robots to reduce onboard computation but do not analyze latency or accuracy impacts. In Visual SLAM (VSLAM) applications, studies [2, 11] highlight offloading's performance benefits for resource-constrained robots, but caution against using it for localization due to network latency concerns. None of these studies apply offloading techniques to resource-constrained autonomous racing platforms where higher speeds impose stricter latency requirements.

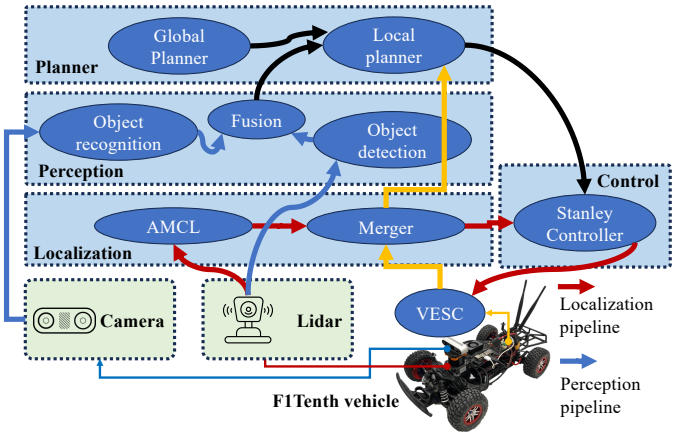


Fig. 2: Overview of F1Tenth racing software stack

III. Background and System Model

A. F1Tenth racing software stack

The software stack of our F1Tenth vehicle, illustrated in Fig. 2, is organized into interdependent layers that support perception, localization, planning, and control. Before each race, we generate a complete map of the track and an optimized global raceline, which initially disregards potential dynamic obstacles. The localization layer, using AMCL, combines LiDAR data with noisy wheel odometry to estimate the vehicle's position relative to the map. This positional estimate is refined through a merger process with wheel speed data, ensuring accurate localization during high-speed maneuvers. In the perception layer, object recognition and detection modules process inputs from both LiDAR and camera sensors, identifying obstacles in the vehicle's path. These detections are further processed through a fusion module to increase the robustness of obstacle recognition. The planning layer consists of a global planner and a local planner: while the global planner provides the initial path, the local planner adjusts it in real-time to avoid obstacles identified by the perception pipeline, which is central to our overtaking approach. Finally, the control layer uses the stanley controller to convert the local planner's output into precise steering and speed commands, which the VESC module translates into physical actions for the vehicle's motors and steering mechanisms.

B. Key performance indicators (KPIs)

We focus on the following key performance indicators (KPIs) that directly influence control quality and racing performance.

End-to-end latency of pipelines. End-to-end latency is defined as the elapsed time from the arrival of sensor data at the first task of a pipeline to the completion of the final task

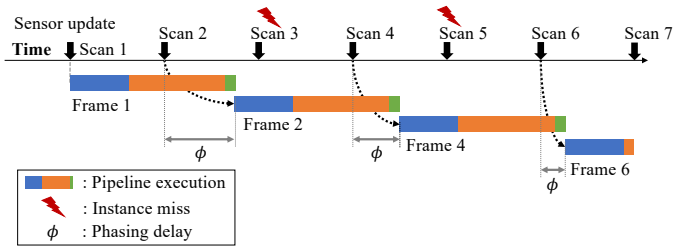


Fig. 3: Phasing delay and scan drops

that produces an updated output. In autonomous racing, this latency directly affects control accuracy and data freshness. For example, latency in the localization pipeline delays pose updates used by the controller, potentially increasing tracking error. Similarly, latency in the perception pipeline can delay obstacle detection, reducing the effectiveness of overtaking or avoidance maneuvers. Therefore, minimizing end-to-end latency of core pipelines is essential for stable high speed operation.

Throughput of pipelines. Throughput represents the number of complete pipeline executions within a given time interval. In the racing context, higher throughput corresponds to more frequent updates of steering and velocity commands. This enables finer-grained control adjustments and smoother trajectory tracking. Reduced throughput, on the other hand, may lead to coarser control updates and degraded performance.

Phasing delay (ϕ). We define *phasing delay* as the time interval between the arrival of new sensor data and the start of its processing within the pipeline. As illustrated in Fig. 3, phasing delay occurs when a newly arrived data instance cannot be processed immediately because a previous instance is still executing.

In the localization pipeline, AMCL combines LiDAR scans with wheel odometry to estimate vehicle pose. However, AMCL processes scans sequentially and employs a recursive mutex within its callback to prevent concurrent execution. If a new scan arrives while the previous scan is still being processed, the new instance must wait until the current execution completes. This serialization introduces phasing delay and eventually causes scan frame drops when accumulated delay exceeds the sensor period. Therefore, it reduces effective throughput and propagate outdated state information to downstream control modules.

Waypoints (w_i) and cross-track error (e_X). The local planner generates a continuous trajectory using splines, which is discretized into waypoints at fixed spatial intervals (e.g., 0.2 m). Each waypoint is defined as:

$$w_i := (s_i, x_i, y_i, \psi_i, \kappa_i, v_i),$$

- s_i (m) is the curvilinear distance along the trajectory,
- (x_i, y_i) (m) are the waypoint coordinates in the map frame,
- ψ_i (rad) is the heading angle,
- κ_i (rad/m) is the curvature, and
- v_i (m/s) is the desired speed.

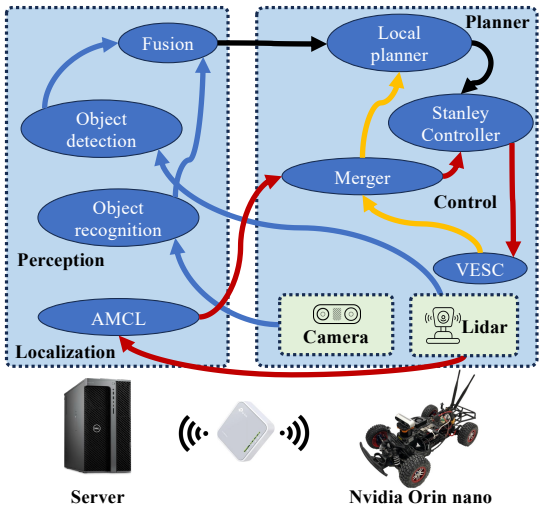


Fig. 4: Task offloading for F1Tenth software stack

To quantify trajectory tracking performance, we use cross-track error (e_X), which measures the lateral deviation of the vehicle from the desired path. It is computed as the projection of the vector from the vehicle position (x_v, y_v) to the nearest waypoint (x_i, y_i) onto the direction perpendicular to the vehicle's heading as follows:

$$e_X = \begin{bmatrix} x_v - x_i \\ y_v - y_i \end{bmatrix} \cdot \begin{bmatrix} \cos(\theta_v - \pi/2) \\ \sin(\theta_v - \pi/2) \end{bmatrix},$$

where θ_v denotes the vehicle's heading angle. The magnitude of e_X reflects how accurately the vehicle follows the planned trajectory.

IV. Architecture of Task Offloading

To reduce latency in computation-intensive pipelines, we design a client-host offloading architecture. As illustrated in Fig. 4, latency-critical tasks such as localization and perception are executed on an external host system, while control remains onboard. This design reduces processing delay on the embedded platform while preserving real-time actuation.

Latency feasibility condition. Offloading improves performance only if the total remote execution time, including network delay, is smaller than the local execution time. For localization, this condition can be expressed as:

$$C_{tr} + \frac{C_A}{f_h} < C_A, \tag{1}$$

where C_{tr} is the round-trip transmission delay, C_A is the worst-case local execution time of AMCL, and f_h is the host speedup factor. In our system, $C_A = 60.5\,\text{ms}$ and $f_h = 2.95$. The measured round-trip transmission delay is approximately 4 ms over WiFi. Thus,

$$4\,\text{ms} + \frac{60.5\,\text{ms}}{2.95} \approx 24.5\,\text{ms} < 60.5\,\text{ms},$$

confirming that offloading reduces end-to-end latency.

Network setup and stability. Communication between the vehicle and host is realized via ROS 2 over a dedicated local

WiFi network, with sensor data and pose estimates exchanged as standard ROS topics. This design requires no modification to the onboard control stack. The current implementation assumes a stable channel with bounded round-trip transmission delay C_{tr}. Network instability, manifested as jitter, congestion, or packet loss, may cause C_{tr} to exceed the margin established by the feasibility condition in Eq. (1), thereby degrading or eliminating the latency benefit of offloading. Formally characterizing the sensitivity of the feasibility condition to network variability, and designing adaptive fallback policies that revert execution to the onboard platform upon condition violation, remain open problems for future investigation.

V. Evaluation

We evaluate the proposed offloading approach using an F1Tenth vehicle [1], on a custom track set up for our autonomous racing practice.

Experimental setups. We configured a client-host setup in which the client (vehicle) and host (laptop) communicate via a WiFi router. The vehicle operates our racing software stack on an Nvidia Orin Nano, equipped with a 6-core ARM Cortex-A78AE 1.5 GHz CPU, a 1024-core Ampere GPU with 32 Tensor Cores, and 8 GB of memory. The host system is a laptop with an Intel Core i7 2.4 GHz CPU, a GeForce 3060 GPU, and 16 GB of memory. Both systems run identical software environments using ROS 2 Galactic running on Ubuntu 20.04. Communication is facilitated by a TP-Link AC750 router, operating on a 2.4 GHz (300 Mbps) band.

We evaluate our approach by offloading the AMCL task in the localization pipeline, i.e., hot topic pipeline with red in Fig. 2, then apply to the perception pipeline. The evaluation is conducted under varying parameter configurations, specifically by adjusting the number of LiDAR scan beams processed by AMCL. A higher number of beams enables more precise vehicle pose estimation, improving localization accuracy, but at the cost of increased computational demand.

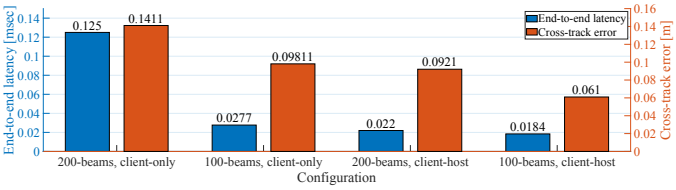


Fig. 5: Comparison of latency and cross-track errors

Comparison of end-to-end latency and cross track error. Fig. 5 shows a comparison of latency and cumulative cross-track error for two LiDAR beam configurations (100 and 200 beams) before and after offloading the AMCL task. Overall, higher latency correlates with increased cross-track error with a strong positive correlation[1] of $r^2 = 0.7223$. As expected, the proposed offloading design significantly reduces end-to-end latency by up to 83% compared to the client-only setup with 200 beams, which, in turn, reduces cross-track error by 35%. The approach also demonstrates its effectiveness with a lower beam configuration (100 beams), achieving a 33.5% reduction in latency and a 37.8% improvement in cross-track error. However, across all configurations, a higher number of beams does not result in lower cross-track error. This is because the increased beam count adds complexity to the processing pipeline, leading to more dropped LiDAR scan frames and higher latency in both client-only and host-client setups.

[1]This r^2 value was the result of a linear regression performed on latency and cross-track error data across both client and client-host setups.

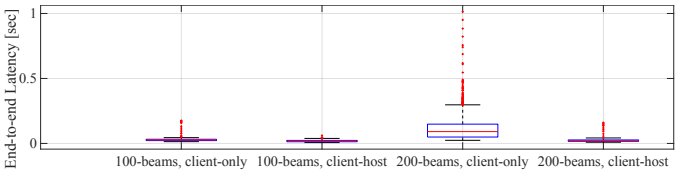
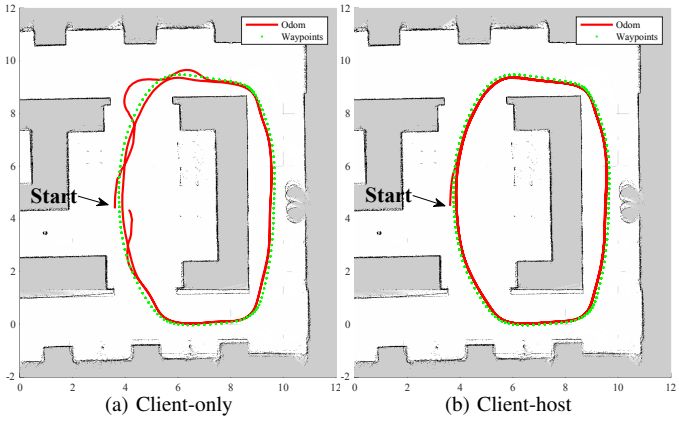


Fig. 6: Latency distribution by configurations

Furthermore, as shown in Fig. 6, the proposed offloading approach produces more consistent latency outputs for the same number of beams, thereby enhancing the vehicle's stability during racing.



(a) Client-only (b) Client-host

Fig. 7: Cross-track error by AMCL offloading

Fig. 7 illustrates the cross-track error results over five laps on our racing track setup. Using the client-host configuration with the 200 beams, the vehicle demonstrates significantly improved navigation of the racing track. In addition to the cross-track error, the average lap time under this configuration improved up to 5.9% compared to the client-only setup, further highlighting the performance gains achieved through our offloading approach.

Comparison of the number of dropped LiDAR scans. In this experiment, we evaluate the number of dropped scan data under varying number of beam configurations. As shown in Fig. 8, our offloading approach proves significantly more effective than the client-only setup in minimizing scan drops. However, these scan drops alone do not fully explain the observed improvements in the above cross-track error. Additional

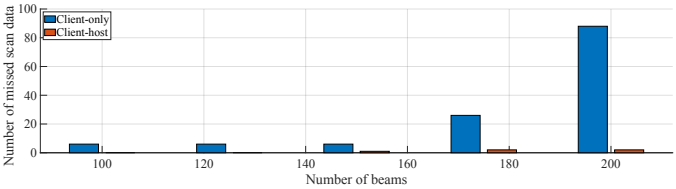


Fig. 8: Missed scan data by configuration

factors, such as waypoint choice error, wheel slip, and random noise, also contribute to the overall performance.

Offloading perception pipeline. We conducted offloading experiments on the perception pipeline, which is one of the critical hot topic pipelines for autonomous racing. We utilized LiDAR-based obstacle recognition and tracking algorithm as described in [9]. The experiments are conducted with varying number of obstacles. Fig. 9 shows the average end-to-end

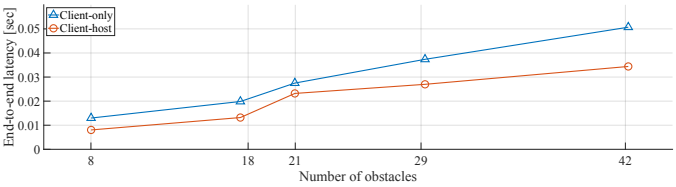


Fig. 9: End-to-end latency of perception pipeline

latency of the perception pipeline. Our offloading approach achieves a latency reduction of 29.4% on average. To validate the significance of these results, we performed a paired t-test[2] [7], which yields a p-value of 0.009169829, below the 0.01 threshold for statistical significance. Therefore, we conclude that offloading perception tasks significantly reduces the latency of obstacle detection.

VI. Conclusion and Future Work

This paper investigated the effect of end-to-end latency of core pipelines on closed-loop control performance in a resource-constrained autonomous racing platform. Experimental results on a F1Tenth vehicle demonstrate that excessive pipeline latency induces scan loss, stale state propagation, and waypoint mis-selection, which collectively increase cross-track error and degrade tracking performance. To mitigate latency-induced degradation, we designed and implemented a client-host offloading architecture that relocates computation-intensive localization tasks to an external processor. The proposed approach reduces end-to-end latency and stabilizes pipeline timing without modifying the control algorithm, resulting in improved tracking consistency under high-speed operation.

Our current evaluation assumes a stable local WiFi network with consistently low round-trip latency. Under degraded network conditions such as high jitter or packet loss, the communication overhead may increase and potentially violate the feasibility condition, reducing or negating the benefits of offloading. Future work will focus on formally modeling the relationship between pipeline latency and control error, incorporating ground-truth pose measurements for quantitative validation, and developing adaptive offloading strategies that gracefully handle network variability, including fallback mechanisms that revert to onboard computation when network conditions violate the feasibility condition.

[2]A paired t-test enables us to isolate the impact of offloading by controlling for a single variable, such as the number of beams or obstacles. If the resulting p-value is below a specified threshold, we can infer that offloading provides a statistically significant advantage.

References

[1] F1tenth. https://f1tenth.org/, accessed October 2024.

[2] P. Benavidez, M. Muppidi, P. Rad, J. J. Prevost, M. Jamshidi, and L. Brown. Cloud-based realtime robotic visual slam. In *2015 Annual IEEE Systems Conference (SysCon) Proceedings*. IEEE, 2015.

[3] A. Bernardi, G. Brilli, A. Capotondi, A. Marongiu, and P. Burgio. An fpga overlay for efficient real-time localization in 1/10th scale autonomous vehicles. In *DATE*. IEEE, 2022.

[4] A. Charroud, K. E. Moutaouakil, and A. Yahyaouy. Fast and accurate localization and mapping method for self-driving vehicles based on a modified clustering particle filter. *Multimedia Tools and Applications*, 2023.

[5] H. Choi, Y. Xiang, and H. Kim. Picas: New design of priority-driven chain-aware scheduling for ros2. In *RTAS*. IEEE, 2021.

[6] F. Gavioli, G. Brilli, P. Burgio, and D. Bertozzi. Adaptive localization for autonomous racing vehicles with resource-constrained embedded platforms. In *DATE*. IEEE, 2024.

[7] G. Geher and S. Hall. *Straightforward statistics: Understanding the tools of research*. Oxford University Press, USA, 2014.

[8] M. Gu, K. Guo, W. Wang, Y. Wang, and H. Yang. An fpga-based real-time simultaneous localization and mapping system. In *2015 International Conference on Field Programmable Technology (FPT)*. IEEE, 2015.

[9] K. Konstantinidis. Development of a detection and tracking of moving vehicles system for 2d lidar sensors. Master's thesis, TU Delft, 2020.

[10] J. Ren, G. Yu, Y. Cai, and Y. He. Latency optimization for resource allocation in mobile-edge computation offloading. *IEEE Transactions on Wireless Communications*, 2018.

[11] L. Riazuelo, J. Civera, and J. M. Montiel. C2tam: A cloud framework for cooperative tracking and mapping. *Robotics and Autonomous Systems*, 2014.

[12] V. K. Sarker, J. P. Queralta, T. N. Gia, H. Tenhunen, and T. Westerlund. Offloading slam for indoor mobile robots with edge-fog-cloud computing. In *2019 1st international conference on advances in science, engineering and robotics technology (ICASERT)*. IEEE, 2019.

[13] C. Shu, Z. Zhao, Y. Han, G. Min, and H. Duan. Multi-user offloading for edge computing networks: A dependency-aware and latency-optimal approach. *IEEE Internet of Things Journal*, 2019.

[14] X. Zhang and Z. Stiltner. Feasibility and reliability of peercloud in vehicular networks: A comprehensive study. *Pervasive and Mobile Computing*, 2024.