Vehicular Micro Clouds for Distributed Computing in the Edge-Cloud-Continuum

Ziqi Zhou, Agon Memedi, Chunghan Lee, Seyhan Ucar, Onur Altintas, and Falko Dressler

Abstract Many applications have been conceptualized in the field of vehicular networking that inherently build upon extensive computing capabilities. Early adoption was entirely cloud based, and many car makers worldwide established their own cloud infrastructure to support this. This was then complemented by cloud-based solutions from third parties. Applications range from entertainment to cooperative awareness to collective perception to cooperative driving. Given the potentially huge delay such applications may experience, as well as to improve the resilience of collaborative applications, the edge-cloud-continuum has been explored from a vehicular perspective. Most notably, the concept of vehicular micro clouds (VMCs) has been introduced, bridging the gap between local processing and delay-extensive cloud solutions. Conceptually, this idea extends 5G multi-access edge computing (MEC) to a distributed computing environment. In this chapter, we review the motivation and underlying architecture of such VMCs. We particularly focus on recent research findings related to task offloading and information up-/downloading from cars to edge to cloud. We conclude

Z. Zhou
School of Electrical Engineering and Computer Science, TU Berlin, Germany, e-mail: zhou@ccs-labs.org

A.Memedi
School of Electrical Engineering and Computer Science, TU Berlin, Germany, e-mail: memedi@ccs-labs.org

C. Lee
InfoTech Labs, Toyota Motor North America R&D, CA, U.S.A., e-mail: chunghan.lee1@toyota.com

S. Ucar
InfoTech Labs, Toyota Motor North America R&D, CA, U.S.A., e-mail: seyhan.ucar@toyota.com

O. Altintas
InfoTech Labs, Toyota Motor North America R&D, CA, U.S.A., e-mail: onur.altintas@toyota.com

F. Dressler
School of Electrical Engineering and Computer Science, TU Berlin, Germany, e-mail: dressler@ccs-labs.org

the discussion with an outlook to ongoing research on distributed artificial intelligence by and for vehicles optimizing driving decisions and improving safety on the road.

Keywords Edge-cloud-continuum · vehicular micro cloud · distributed computing · virtualized edge computing · vehicle to everything communication

1 Introduction

Vehicular networking is the technological basis for many new collaborative applications in the field of intelligent transportation systems (ITS) [1, 2]. Applications range from early cooperative awareness solutions to collaborative perception and to cooperative maneuver management. The technological basis is provided by vehicle to everything (V2X) communication protocols such as dedicated short range communication (DSRC) and cellular V2X (C-V2X) [3]. Such communication is an enabler for a multitude of applications. At the time being, mostly 5G-based communication is used in combination with cloud computing-based applications. Almost every car maker provides their own cloud solutions for enhancing the driving experience. Following this line of thought, cooperative coordination among cars is very complicated as these cloud solutions are (1) closed islands of single car makers and (2) cloud solutions are strictly limited by available communication resources from the car to the mobile network and to the cloud provider. Multi-access edge computing (MEC) is considered to help break these barriers by providing local computing resources and bridging gaps between providers [4, 5]. At the same time, most relevant sensor data is generated locally, often at very high data rates (e.g., lidar and HD camera sensors). Sending this data uncompressed to some edge server is prohibitive, given the limited wireless communication resources. Thus, local (pre-)processing also needs to be included in this perspective. This becomes even more relevant considering the upcoming AI/ML applications for cooperative vehicular applications [6]. Taking all these concepts into consideration, the so-called edge-cloud-continuum [7, 8] aims at integrating these approaches and providing optimized ways for resource management and distributed computing – of course, also in vehicular applications.

In this chapter, we review concepts of this edge-cloud-continuum, focusing on the vehicular use case. We study how edge computing can help optimize resource utilization and how edge computing can be directly integrated in this context. The fundamental basis for distributed computing in this application domain was established back in 2012 with the conceptualization of vehicular cloud computing [9, 10]. Meanwhile, with the upcoming 6G standards, edge computing is getting more generalized, also because 5G MEC has not yet been deployed at a large scale. The focus is mostly on integrating local computing capabilities into the edge-cloud-continuum. A milestone in this direction has been the formulation of the virtualized edge computing (V-Edge) concept [11]. MEC reduces this delay by placing (or just identifying) computation resources close to users. The concept of a vehicular micro cloud (VMC) extends this idea by turning nearby vehicles themselves into temporary virtual edge servers [12, 13]. In the scope of this chapter, we review the concept of a VMC focusing on how such VMCs are

established and maintained, given the fact that vehicles induce fast topology changes. We then revisit two core features of such VMCs, namely keeping data ready for use in close proximity to the user, as well as providing distributed computing capabilities for task offloading.

Organization of this chapter: We first investigate the concept of a vehicular micro cloud in Section 2. This includes the system model and a feasibility study. In Section 3, we review algorithms for using VMCs for distributed data storage. The focus is on data retention and recovery. In Section 4, we go beyond and illustrate how efficient data up- and downloading to/from a VMC can be realized. In Section 5, we study computational offloading in VMCs. Such offloading allows us to realize task offloading and scheduling in VMCs. Finally, a framework for developing new DRL-based offloading concepts is introduced in Section 6.

2 Vehicular Micro Clouds

This section is based on [14, 13]. In short, a vehicular micro cloud (VMC) is a group of vehicles in proximity that cooperate to form a small temporary cluster, representing a virtual edge server [13]. Modern cars have sensors, processing units, and wireless interfaces; by using them, vehicles can perform local data sharing, caching, and computation without relying on fixed infrastructure [14]. This section introduces the concept of VMCs and explains how it extends traditional edge computing paradigms to highly dynamic vehicular environments.

2.1 System Model

Vehicles first form a VMC, e.g., around a certain geographical location. An example is depicted in Figure 1. In the figure, intersections are used as landmarks for VMC formation. Of course, also moving VMCs are possible, e.g., on a freeway. In the scope of this chapter, we focus, without loss of generality, on stationary ones. In order to manage a VMC, clustering concepts are often used to select all vehicles in a geographical space as members. In order to simplify management, one cloud member (CM) is identified for coordination as a cloud leader (CL). The CL manages all operations in its VMC. Data can be stored using methods like distributed hash tables, and when a car leaves, its data is handed over to another member. Figure 1 also shows a second VMC. Obviously, handover mechanisms are required for such multi-VMC operation. In the following, we will explore in more detail how to set up and manage VMCs. The figure also includes an access point (AP), e.g., a road side unit (RSU) or a base station (BS). Such infrastructure elements can be used for improved coordination of VMCs.

VMCs can be classified according to three main aspects: *parked* or *moving* car members, *stationary* or *mobile* regions, *preassigned* or *on-demand* VMCs. If *parked* cars are part of a VMC [15, 10], this helps make operation rather robust given the long

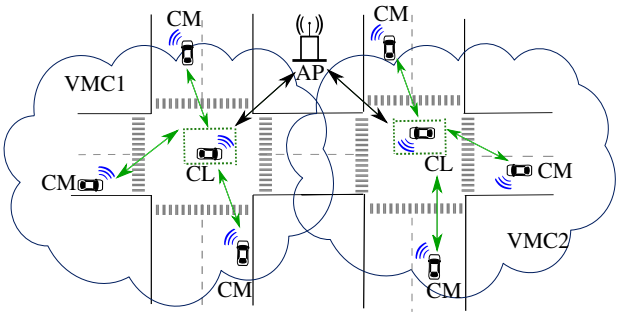


Fig. 1: VMC system model: Geolocated VMC indicating VMC members as cluster members (CM) and cluster leader (CL) [13]. The key aspect is the dynamic formation of a VMC, where vehicles in proximity can communicate via V2X communication. Vehicles can join and leave the cluster, while the VMC continues to operate as a functional system.

dwell times (i.e., the time the vehicle stays within the VMC). *Moving* cars are probably the more general case. Here, cars are temporarily joining the VMC, participating in data storage and computation-sharing, and leaving again. This will also be our main focus in the following. VMCs can also be classified into *stationary* or *mobile* ones. *Stationary* VMCs are formed at fixed geographic areas such as intersections. *Mobile* VMCs are clouds formed by vehicles traveling close to each other, e.g., on a freeway. Finally, VMCs can be formed in a *pre-assigned* or an *on-demand* manner. The *pre-assigned* approach requires a central controller running on an AP. *On-demand* VMCs are created by the vehicles themselves in an ad-hoc manner.

2.2 Macro-Micro-Cloud Architecture

The macro-micro-cloud architecture [13] extends the idea of VMC by organizing and allocating computation and storage across three layers: the macro cloud, the micro cloud, and end users. The three layers form a hierarchical system that facilitates information communication, decision making, task processing, and data management in vehicular clusters. The micro cloud layer is built from groups of vehicles forming VMCs. The macro cloud interconnects multiple VMCs and can be integrated with backend data centers that perform large-scale data processing and store long-term data. The architecture also covers end users that are not directly part of the VMC architecture. Such end users can be individual vehicles, pedestrians, and roadside sensors. They use the VMC in a way similar to MEC for offloading tasks or exchange of information. When tasks are executed or data is managed by such virtual edge servers, the response time becomes significantly faster compared to the cloud center.

2.3 Feasibility

After introducing the concept of VMCs, we need to study the feasibility in the real world. Simulations of Higuchi et al. [12] show that under many circumstances, vehicles can find or form VMCs. In this paper, the feasibility of VMCs models was explored using SUMO[1] simulations, including stationary VMCs, mobile VMCs, and the connection between VMCs. Results for the city of Luxembourg using the LuST model [16] show that with a 300 m radius per VMC, over 97% of cars found a stationary VMC within 1 km. As for mobile and on-demand VMCs, about 62% of vehicles found the nearest micro cloud within 1 km. Also, 90% of the VMCs could communicate with another VMC within 1 km, showing that macro-micro-cloud systems can also be formed. The study also reveals that the successful formation of a VMC highly depends on concrete traffic conditions. For example, during off-peak hours, few stationary clouds are formed.

2.4 VMC Formation Steps

Hagenauer et al. [14] proposed a map-based stationary VMC design. The VMCs are formed at intersections where cars from different directions typically maintain strong line of sight connectivity. The original formation procedure includes four steps.

- Vehicles periodically broadcast control information.
- Vehicles locally detect clusters based on distance, connectivity, or map-based rules (e.g., proximity to an intersection).
- A CL is elected using a distributed rule.
- The CL announces its role, and surrounding vehicles join as CMs.

2.5 Dwell Time Prediction in VMCs

Dwell time refers to how long a vehicle stays inside a VMC. This duration directly affects whether a task or data transmission can be completed successfully. If either the sender or the receiver leaves the VMC too early, the operation fails. For this reason, estimating the dwell time is essential for reliable task offloading, data exchange, collaborative up-/down-loading, and data recovery. Pannu et al. [17] established an empirical model for dwell time prediction based on mobility data for the city of Luxembourg. The model gives very accurate results for all major intersections of the city. Unfortunately, it cannot be generalized to other places. Schettler et al. [18] therefore developed a reinforcement learning based approach to overcome this limitation. The resulting model outperforms the heuristic and is able to adapt itself to other city layouts.

In most scenarios, it is unnecessary to predict the entire time a vehicle will remain in a VMC. If only time-sensitive tasks and data will be shared and processed in VMC

[1] https://sumo.dlr.de/docs/index.html

scenarios, we only need to know whether a vehicle will stay long enough to finish the current task. To support this, Zhou et al. [19] proposed a new machine learning-based upper-bounded dwell time prediction model designed for reliable task offloading. The model applies a Random Forest predictor due to its robustness and ability to capture nonlinear driving behavior. The model was validated using mobility traces from Luxembourg [16] and Nagoya [20].

2.6 Applications of VMCs

VMCs enables a wide range of applications by leveraging temporary connected vehicle clustering in a certain area where vehicles share metadata and collaboratively process data. The primary applications include data management within a VMC, data recovery across multiple VMCs, collaborative data uploading or downloading facilitated by VMCs, and task offloading within a VMC. Selected applications and algorithmic solutions will be discussed in the following sections.

Data Management: Data is stored and indexed using various protocols across VMC members. Cloud members and leaders store different data segments based on storage decisions, which can be determined by centralized controllers or through ad-hoc coordination. A practical application is map caching, where large maps are fragmented and strategically distributed across vehicles within the same VMC. The VMC maintains a lookup table, enabling new vehicles to efficiently download required segments from neighboring cars rather than accessing distant cloud servers.

Data Retention and Recovery: Given the high mobility of vehicles, VMC membership changes frequently. A relevant question is to what extent data can be kept persistent using replication from departing vehicles to new or remaining members [21]. When data loss occurs, such as when all vehicles containing specific fragments leave simultaneously, recovery protocols reconstruct missing chunks using erasure coding with remaining fragments or fetch them from cloud storage. A concrete example is live traffic video streaming at critical intersections, where the system maintains continuous video feeds despite vehicle departures. The recovery protocol executes three steps: (1) requesting data from other members, (2) reconstructing using network coding, and (3) fetching from cloud servers as a last resort.

Collaborative Data Upload: Instead of individual vehicles struggling with weak cellular connections to upload large datasets, e.g., traffic camera footage or map updates, VMCs enable efficient distribution of encoded chunks via vehicle to vehicle (V2V) communication. Vehicles can then upload these chunks when connected to Wi-Fi, significantly reducing cellular costs and improving reliability [22].

Collaborative Data Download: Similarly, large files such as map updates can be divided into chunks distributed among multiple vehicles. After downloading chunks, vehicles share them via V2V communication, enabling all VMC members to access complete files much faster than through individual downloads [23].

Computational Task Offloading: VMCs leverage collective computing resources to perform complex tasks. Vehicles can offload computationally intensive operations, such as real-time object detection or camera feed analysis, to idle or high-capacity neigh-

boring vehicles. This enables faster processing by utilizing distributed computational power within the VMC.

2.7 Summary

In summary, VMCs provide a flexible framework for utilizing the collective resources of vehicles to support distributed computing at the virtual edge. By dynamically forming clusters of vehicles, VMCs enable virtual edge processing, within-VMC storage, collaborative uploading and downloading, and various additional functions. This concept can enable the mechanisms discussed in the following sections, including distributed storage, collaborative data transfer, and task offloading.

3 Using a VMC as Distributed Storage

This section is based on [21, 24]. In this section, we discuss how distributed storage can be realized within VMCs. Due to the highly dynamic nature of vehicular networks, storing and maintaining data across multiple vehicles in the same VMC requires reliable mechanisms regardless of frequent traffic changes. Therefore, we introduce approaches that enable redundancy, replication, and coded storage across vehicles to maintain data availability.

Stationary VMCs typically rely on clustering mechanisms to group nearby vehicles. Each data item generated or stored within a cluster is associated with that specific *parent* VMC. Vehicle mobility, however, poses a threat to data availability: cars join the cluster, hold data for only a short time, and then leave the region. If the departing vehicle carries the last remaining copy of a data item, that item becomes irretrievably lost at its geographic origin. In small or fluctuating cluster sizes, common in low and medium traffic densities, the data are more likely to have a single copy, therefore being highly vulnerable to loss. Even in dense traffic, rapid variations can temporarily empty an intersection in a short time, wiping out all locally stored data if no recovery mechanism exists. As a result, extreme node churn, short dwell times, and volatile traffic patterns render traditional replication strategies insufficient. The core challenge, therefore, is data retention: ensuring that locally relevant information remains continuously available within its parent VMC despite the constant turnover of vehicles.

3.1 The Mobility Challenge: Data Loss in VMCs

One way to address the challenge of retaining data in VMCs is to exploit vehicular mobility itself, by treating vehicle trajectories as an implicit means of transport for restoring lost or departing data to its geographic origin. The intuition behind this mechanism is depicted in Figure 2:

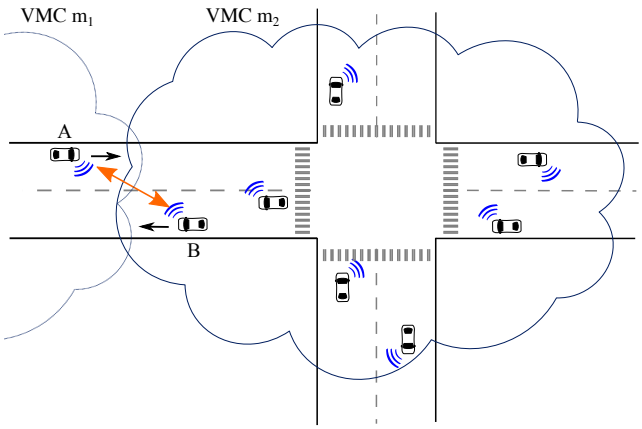


Fig. 2: Typical data return scenario in a VMC [21]. Mobility itself can be an assistant for restoring data to the original VMC. In this show example, vehicles moving between neighboring VMCs act as carriers that transfer data items or tasks across clusters, allowing data to return to its parent VMC even if the original holder leaves the VMC.

- Vehicle A is about to leave VMC m_1 while holding a copy of data item d associated with m_1.
- Vehicle B is currently in a neighboring VMC m_2 but is predicted to enter m_1 within the next few seconds based on its navigation trajectory.
- Before leaving communication range, A transfers data item d to B.
- When B enters m_1, d is effectively returned to its parent VMC m_1.

Repeating this handoff across adjacent VMCs creates a cooperative data-return strategy that improves geographic data persistence. To leverage this behavior, a mobility-aware protocol was introduced [21] that coordinates data exchanges between neighboring VMCs and restores items that would otherwise be lost due to rapid turnover. Using vehicle mobility as the core retention mechanism, the protocol addresses one of the key challenges of VMCs: maintaining data availability in highly dynamic environments.

The data return protocol involves two distinct types of data exchange necessary for distributed data recovery: *control beacons* and *VMC data* transmissions.

Control Beacons for Information Exchange: Every car periodically broadcasts control beacons. These beacons serve as a mechanism to inform surrounding vehicles of the current state of the car and its data requirements. Each beacon includes four essential pieces of information:

1. The sender car's identifier (ID).
2. The current VMC ID to which the sender belongs (parent VMC).
3. A set of data contents the car currently holds.
4. A set of missing data content IDs, that the car desires to access.

Microcloud Metadata Table – The Local Knowledge Base: Every vehicle continuously receives short control beacons from all neighbors within radio range. Rather than

treating these beacons as transient advertisements, each vehicle aggregates them into a small local database called the VMC metadata table (T). The purpose of this table is to provide each vehicle with a concise overview of:

1. The data items it currently holds, and
2. The data items that nearby vehicles hold or are missing.

Table 1 shows a concrete example maintained by vehicle c_1. The crucial row is the last one: vehicle c_4 is still physically in VMC m_2, but its navigation route tells it that m_1 is its next (or next-next) stop. Therefore, c_4 has already begun announcing interest in m_1's data (d_1 and d_2), even though it is not yet inside m_1. This early announcement is what enables the hand-off: departing vehicles in m_1 see c_4's δ_T and realize "c_4 is the perfect vehicle to bring my data back soon."

Table 1: Example VMC metadata table maintained by vehicle c_1.

Vehicle	Current MC	Data held (item, parent)	δ_T (items wanted)
c_1 (self)	m_1	$\{d_1, m_1\}, \{d_2, m_1\}$	$\{d_3, m_1\}$
c_2	m_1	$\{d_1, m_1\}$	$\{d_2, m_1\}, \{d_3, m_1\}$
c_3	m_1	$\{d_2, m_1\}, \{d_3, m_1\}$	$\{d_1, m_1\}$
c_4	m_2	$\{d_8, m_2\}$	$\{d_1, m_1\}, \{d_2, m_1\}$

What Do I Need? – Determining a Vehicle's Missing Data Set (δ_T): As a vehicle moves, it is expected to enter a sequence of VMCs along its route. Let M denote the ordered set of these upcoming VMCs. Each VMC is associated with specific data items that should ideally be available within it. Using the metadata table T, a vehicle compares:

- The data items expected to exist in the upcoming VMCs (based on entries reported from nearby vehicles), with
- The data items the vehicle currently possesses.

The result of this comparison is the vehicle's *missing-data set*, denoted δ_T. Intuitively, δ_T contains all data items that the vehicle does not have, but that should exist in one of the VMCs it will soon enter. Formally, δ_T is computed by subtracting the set of data items owned by the vehicle from the set of data items known (through T) to be associated with the upcoming VMCs. Accordingly, δ_T is broadcast in the next beacon, signaling exactly what the vehicle is willing to accept right now.

What Can I Offer? – Selecting the Data Item for Transmission: After determining which data items are missing, a vehicle must also identify which data items it can provide to others. This is captured by the set Δ_{Tx}, which consists of the data items that:

- Are currently possessed by the vehicle, and
- are explicitly requested by at least one neighbor who will soon enter the corresponding parent VMC.

Thus, Δ_{Tx} represents the subset of data items that would be useful to forward to support the availability of data throughout the system. Whenever Δ_{Tx} is non-empty, the vehicle

selects one data item d_{Tx} for transmission. By default, it chooses the item with the longest remaining lifetime. This greedy strategy aims to maximize the future utility of each transmission. To reduce redundant transmissions and avoid unnecessary channel load, a cooperative suppression strategy is applied: Every receiver that obtains d_{Tx} instantly removes that item from its own Δ_{Tx}. This prevents the vehicle from requesting or retransmitting the same item again.

3.2 Adaptive Mechanisms for Efficient Channel Utilization

A key challenge for our VMC data return protocol is balancing two conflicting objectives: on the one hand, maximizing data recovery across neighboring VMCs and, on the other hand, preventing excessive wireless channel usage. If vehicles request data for too many upcoming VMCs (denoted by n) the resulting increase in messaging can overload the channel. To manage this trade-off, the protocol integrates adaptive mechanisms that modulate communication behavior according to current channel conditions.

Adaptive Selection of Next Microclouds (n): Each vehicle periodically measures the current channel utilization, denoted by γ, using the standard carrier-sensing mechanism. Channel utilization is represented as the fraction of time during which the channel is sensed as busy. Based on this measurement, the vehicle determines the number of upcoming VMCs for which it will request data. Let N denote the maximum number of VMCs a vehicle is allowed to consider. The effective value of n is then computed as:

$$n = \begin{cases} N - \lfloor \gamma \times \omega \rfloor, & \text{if } N > \lfloor \gamma \times \omega \rfloor, \\ 1, & \text{if } N \leq \lfloor \gamma \times \omega \rfloor, \end{cases} \tag{5}$$

where ω is a scaling factor (set to 10 in our case). This adaptive rule reduces n when channel utilization increases, thereby limiting the number of data requests a vehicle generates. For very high channel load, the vehicle restricts its focus to the current and immediate next VMC, preventing further congestion.

Adaptive Data Transmission Interval: Beyond adjusting n, the protocol also adapts the data transmission interval to further mitigate channel saturation. This is achieved through an adaptive transmission window, denoted by $\mathrm{Tx}_{\mathrm{window}}$:

$$\mathrm{Tx}_{\mathrm{window}} = [0,\ \max(1, \lfloor \gamma \times \omega \rfloor)]. \tag{6}$$

For each transmission opportunity, the vehicle selects a random value from this window as the next transmission interval. As channel utilization increases, the window expands, effectively lengthening the average interval between transmissions. This mechanism helps prevent overload while maintaining regular communication in low-traffic periods.

Performance Implications for Data Availability: The performance of the VMC data return protocol is determined by its ability to counteract data loss caused by high vehicular churn, while simultaneously operating within the constraints of limited wireless resources. A systematic evaluation of the protocol demonstrates both its necessity and

its effectiveness across a wide range of mobility and channel conditions. A detailed description of the simulation setup and parameters can be found in [21].

A principal challenge arises from the short duration during which a vehicle remains within the coverage of a stationary VMC. All data replication and recovery operations must therefore be completed within this time window; otherwise, the parent VMC risks losing its data entirely. The evaluation environment (Manhattan grid) included multiple vehicle densities (50, 100, and 200 vehicles) and several data sizes ranging from 2 kB to 10 kB. These variations resulted in average channel utilization between approximately 5% and 70%. As a result, the protocol was examined under both lightly loaded and highly congested wireless conditions, providing a comprehensive understanding of its behavior.

In summary, the protocol ensures that data availability is maintained for a significantly longer duration, even in highly mobile environments, thereby allowing VMCs to serve as a reliable platform for critical ITS applications.

3.3 Data in VMCs – Coded Caching for Efficient Access

Ensuring that data remains available inside a VMC is one step toward enabling dependable edge services; Once data fragments are retained within the VMC, another challenge is how vehicles efficiently access those distributed contents. The most significant challenge stems from vehicles frequently joining and leaving VMCs due to traffic flow, leading to a highly dynamic topology. Consequently, the distributed resource pool, including cached data, is constantly changing. Under these stringent constraints, ensuring efficient access to the data stored by mobile nodes, becomes essential for the proper functioning of the system.

To address this, Pannu et al. [24] introduced a coded caching-based data distribution protocol that significantly reduces channel load and accelerates data access in VMCs. The method adapts classic coded caching principles to the constraints of mobile, decentralized vehicular environments, where vehicles act simultaneously as consumers and data providers. Coded caching is an information-theoretic concept designed to reduce network load and improve latency over shared wireless media [25]. The approach divides each data object into smaller fragments and operates in two fundamental phases:

1. Partial Content Caching: Fragments of data contents are stored in the local cache of end-users (in this case vehicles) during periods of low network load.
2. Coded Multicasts: When requests arrive during high network load, encoded messages are broadcast to serve multiple unique data content requests simultaneously, thereby saving bandwidth.

The VMC setting introduces characteristics that differ from conventional coded caching scenarios. Vehicles act concurrently as *clients*, holding partial fragments in their local caches, and as *servers*, supplying those fragments to other VMC members. When a coded transmission is broadcast, any vehicle that already stores one of the encoded fragments can decode the missing fragment. Consequently, a single coded

multicast enables several vehicles to recover different requested data items, effectively mitigating channel congestion and improving overall system efficiency.

The implementation of coded caching yields significant efficiency gains in VMC operations, namely:

- Local Gain: This gain is derived simply from having data contents already cached locally, reducing the need for any transmission.
- Global Caching Gain: This gain is the benefit of encoded broadcasts. Since the transmission is coded based on the contents available in the caches of other users, a single transmission serves several requests. This process reduces the total number of transmissions required, significantly reducing the average channel load.

By exploiting the overlap between cached and missing fragments across vehicles, the system encodes multiple fragments into a single transmission. Each vehicle that possesses at least one of the encoded fragments can decode the missing part using simple XOR operations. The efficiency of data sharing within the VMC can be improved by up to 50% through this technique, making it beneficial for high-mobility environments where bandwidth is scarce.

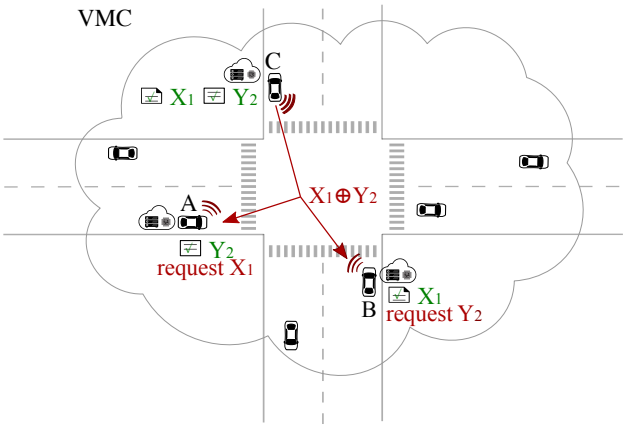


Fig. 3: Example scenario showing a virtual edge at an intersection [24]. Vehicles in the VMC collectively store different data fragments and exploit the XOR features to serve multiple requests with a single coded transmission. The figure focuses on how the vehicle holding both fragments required generates an encoded chunk that allows multiple vehicles to recover one missing chunk simultaneously.

Example of Coded Caching in a Vehicular Microcloud: To illustrate the operation of coded caching in a VMC, consider the scenario shown in Figure 3, where a VMC is formed at a road intersection. Three vehicles participate in the VMC and collectively store different fragments of two data contents, denoted X_1 and Y_2, as shown in Table 2.

Without coded caching, these requests would be satisfied through two separate transmissions: (a) Vehicle A sends Y_2 to Vehicle B, while Vehicle B sends X_1 to

Table 2: Example of the use of coded caching.

Vehicle	**Stores**	**Requests**
Vehicle A	Y_2	X_1
Vehicle B	X_1	Y_2
Vehicle C	X_1, Y_2	$\varnothing$

Vehicle A; (b) Vehicle C sends X_1 to A and Y_2 to B. In both cases, two wireless transmissions are required.

With coded caching, Vehicle C can satisfy both requests using a *single broadcast*. Instead of sending the fragments individually, it generates an encoded packet using the XOR operation: $X_1 \oplus Y_2$. This coded packet is broadcast once to all VMC members. Each receiving vehicle can then recover the missing fragment using the fragment it already stores. Specifically,

$$\text{Vehicle A:} \quad X_1 = (X_1 \oplus Y_2) \oplus Y_2, \qquad \text{Vehicle B:} \quad Y_2 = (X_1 \oplus Y_2) \oplus X_1.$$

As a result, both vehicles decode the requested data from a single transmission.

Protocol Operation: The protocol relies on robust information exchange among VMC members. Each vehicle maintains a *virtual edge management database* storing information about all other participating members. *Control beacons* are periodically transmitted by every vehicle. These beacons contain two key lists of information:

1. *Available Data IDs*: A list of data fragments currently cached by the vehicle.
2. *Missing Data IDs (Requests)*: A list of data fragments the vehicle needs to access.

Upon receiving a beacon, a vehicle updates its virtual edge management database with the sender's information. This ensures that all VMC members share a consistent view of cached data and active requests.

Identifying Encoding Opportunities: The core challenge of coded caching is determining which data contents can be efficiently encoded together to satisfy multiple pending requests in a single transmission. To address this, each vehicle periodically executes the following internal process:

1. *Candidate Duplet Calculation*: The vehicle pre-calculates all possible data duplets (pairs of fragments, e.g., k_i, k_j) from its own local cache.
2. *Request Matching*: The vehicle searches for data duplets, (k_i, k_j), that satisfy the *reciprocal store and request condition* among peer vehicles. For example: Vehicle V_1 stores k_i and requests k_j, and Vehicle V_2 stores k_j and request k_i.
3. *Result Compilation*: The algorithm compiles a list of all potential data duplets that can be encoded and transmitted to satisfy multiple concurrent data requests.

The Encoding and Transmission Process: Once the candidate list is prepared, the transmitting vehicle must select which duplet to use. Various heuristics may be applied. One possible strategy is to select the duplet with the maximum average time passed since the data requests were initially made, ensuring requests do not "starve." The selected data fragments are encoded using a bitwise XOR operation. The size of this

resulting encoded message remains equal to the size of a single partial data content, achieving significant bandwidth efficiency.

Decoding and Cache Updates Upon receiving the encoded data, a vehicle follows a strict procedure:

1. *Decodability Check*: The vehicle determines if the received data is decodable. This is possible if one of the fragments encoded in the message is already present in the vehicle's local cache.
2. *Decoding*: If decodable, the vehicle uses its cached fragment to XOR with the received encoded data, thereby retrieving the missing fragment (e.g., $X_1 = (X_1 \oplus Y_2) \oplus Y_2$).
3. *Saving Encoded Data*: If the data is not immediately decodable, the vehicle stores the raw encoded data in its cache, as it may be used later to decode other data contents if needed.

As vehicles successfully receive and cache new data contents, they update their lists of required and available data contents. These updated lists are then broadcast in subsequent control beacons, ensuring the virtual edge management database remains synchronized. Requests that have been fully served are subsequently removed from the management table to prevent unnecessary repetitive transmissions.

The data access protocol based on coded caching is intentionally designed to be structurally similar to the data recovery protocol discussed previously. This allows the two protocols to operate effectively in combination for efficient virtual edge operations. In essence, the data recovery mechanism ensures that the content remains available within the VMC, while the coded caching mechanism ensures that the available content is accessed efficiently by VMC members. When the coded caching protocol operates alongside the trajectory-aware data return protocol, the virtual edge can both preserve data fragments that would otherwise be lost and distribute them efficiently using a minimal number of wireless transmissions.

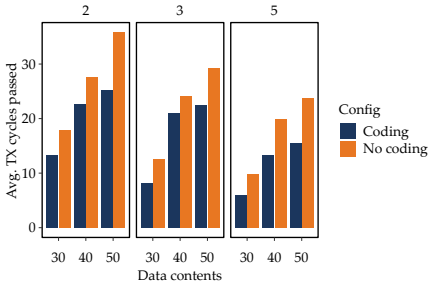
Performance Implications of Data Sharing using Coded Caching: The performance of the coded caching protocol is determined by its capacity to reduce wireless channel load while preserving timely and reliable data access within the highly dynamic VMC environment. By exploiting cache diversity and serving multiple independent requests through a single coded transmission, the protocol alleviates contention on the shared spectrum and mitigates delays inherent to frequent data exchanges.

For performance evaluation, the following two configurations were compared:

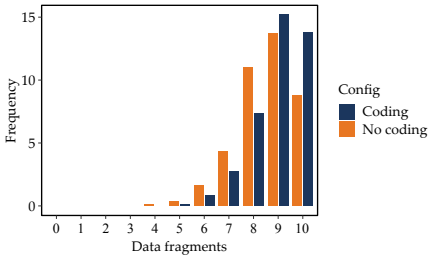
- *No coding (baseline):* Vehicles serve requests by transmitting individual fragments without coding. The fragment with the longest pending request is prioritized.
- *Coded caching:* Two data contents are XOR-encoded following the procedure described above. This enables a single broadcast to satisfy multiple requests.

A detailed description of the simulation setup and parameters can be found in [24].

Simulation results demonstrate that the coded caching approach consistently achieves significantly lower average access times than the no-coding baseline (see Figure 4a). This improvement stems from the multicast nature of coded transmissions: a single broadcast can satisfy multiple independent requests, thereby reducing the total number of transmissions required.



(a) Average number of data transmit cycles until a complete data content is available in the cache of a vehicle. Data is shown for a vehicle density of 20 vehicles. The labels on top of facets represent the data redundancy factor.



(b) Histogram of successfully received data fragments for a redundancy factor of 5 and a vehicle density of 20 vehicles in the virtual edge.

Fig. 4: Performance gain of coded caching [24].

A data request is considered successful only if a vehicle receives *all* fragments of the requested content. Figure 4b presents the distribution of successfully received fragments per request (without the recovery protocol). The coded caching approach yields a noticeably larger number of complete data objects than the no-coding baseline. The bar corresponding to 10 fragments indicates fully satisfied requests, with no missing pieces. A notable observation is that when coded caching is used the entire distribution shifts toward higher fragment counts, as vehicles are able to recover more of the requested content. This improvement stems from the coding gain: a single short broadcast can simultaneously deliver the missing fragment to multiple vehicles, thereby reducing the overall number of transmissions required within the VMC.

In summary, the application of coded caching in VMCs provides an effective solution to the data-access challenges introduced by rapid mobility. Traditional unicast-based retrieval methods easily overload the shared wireless channel, particularly when multiple requests occur simultaneously. The coded caching protocol mitigates this limitation by using lightweight bitwise XOR operations, allowing a single coded transmission to satisfy multiple independent data requests at once. Performance evaluations show that this approach significantly reduces data access times-often by up to 50% – while improving the overall efficiency of the shared wireless medium.

3.4 Summary

Overall, distributed storage in vehicular micro clouds enables vehicles to cooperatively maintain data despite intermittent connectivity and mobility. Techniques such as replications and coded storage improve data durability and availability while distributing storage overhead across multiple cloud members. These mechanisms play a significant role in supporting reliable data services in highly dynamic vehicular environments.

4 Collaborative Data Up-/Download

This section is based on [22, 26, 27]. During driving, connected vehicles generate a large amount of data from multiple sensors and vision cameras [28]. Moreover, they receive data such as multimedia content and High-Definition (HD) navigation maps from remote servers. Thus, vehicle-to-everything (V2X) communication [29, 30] is vital for data exchange. However, the current upload and download of vehicle data depend on the cellular network, and only a small amount of data is collected due to associated costs. To accelerate data-driven development [31, 32] and reduce cellular costs, it is important to minimize the volume of data transferred over cellular networks. In this section, we introduce the concept of collaborative data upload [22, 26] and download [23] using VMCs.

4.1 Collaborative Data Upload

A collaborative data upload method has been proposed [22, 26] to accelerate data collection and reduce cellular costs. The concept of collaborative upload is shown in Figure 5. A connected vehicle detects an event and records a video (Step-1). It then becomes a VMC leader (CL) and forms a VMC at an intersection. The CL transmits the data to VMC members (CMs) in the vicinity (Step-2). The CMs then upload the video data when they connect to a Wi-Fi network (Step-3).

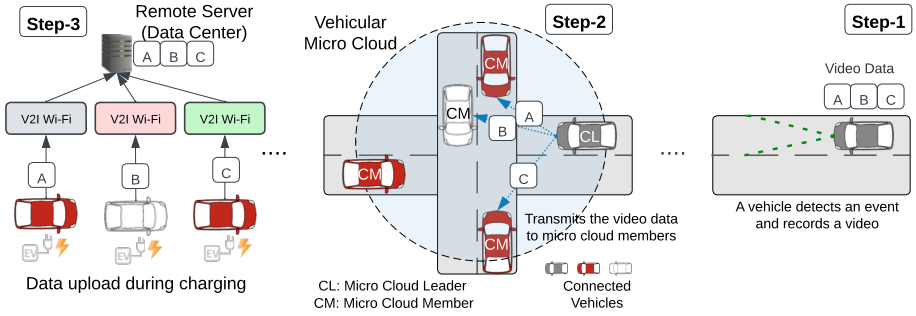


Fig. 5: Concept of collaborative data upload supported by a VMC [22, 26]. The figure highlights how vehicles can use different wireless communication technologies to upload collected events while charging. This requires coordination by a VMC to distribute collected data to vehicles that will (likely) have an uploading opportunity soon.

For a large-scale simulation study, the Los Angeles downtown scenario was used [33]. The simulation parameters are summarized in Table 3. No data prioritization when transferring data to CM vehicles is assumed. The simulation steps are as follows:

1. A vehicle with data becomes a CL and forms a VMC when it encounters CMs.

2. The CL transfers the data to nearby CMs via V2V communication.
3. Once CMs reach their destinations, we assume they connect to a Wi-Fi-enabled charger. Upon connection, CMs upload the stored data via V2I Wi-Fi.

Table 3: Parameters of Los Angeles downtown simulation.

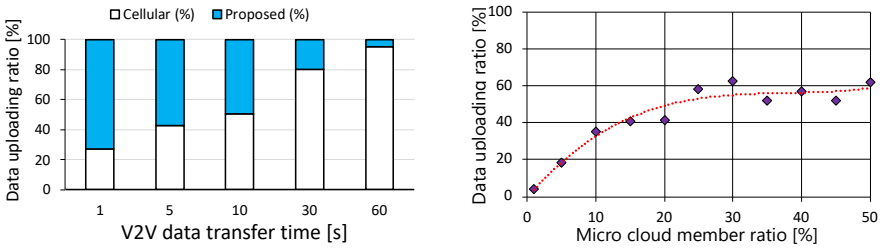
Parameters	Values
Average speed	48.2 [km/h] (about 30 [miles/h])
Average trip length	7.5 [km]
Total vehicles	144K vehicles
Simulation time	6-7 am (morning peak time)
Intersections	459
Connected vehicles for VMC	1 [%]
Video data size	30 [MB]
Storage units	30 [units]
Simulation period	600 [s]
Micro cloud member ratio	$1 \sim 50$ [%]
V2V coverage (diameter)	300 [m]
V2V data transfer time	1,5,10,30,60 [s]

Moreover, two different simulation scenarios were prepared focusing on data transfer time and CM ratio.

- *Controlling V2V data transfer time.* To evaluate how many uploads can be achieved with the proposed method, the V2V data transfer time is varied from 1s to 60s. The CM ratio is fixed at 25% in this simulation.
- *Controlling VMC member ratio.* To show how many CMs are required for the proposed method, the CM ratio is varied from 1% to 50%. In this scenario, the V2V transfer time is fixed at 10s.

Evaluation of collaborative data upload: We focus on the feasibility of collaborative upload through a large-scale urban simulation, evaluating the data upload ratio and the effect of the CM ratio. We present the data upload ratio for different V2V data transfer times in Figure 6a. At short V2V transfer times (1s and 5s), the effect of the proposed method is significant. At a transfer time of 10s, approximately half of the data uploads can be achieved via the proposed method. In contrast, collaborative upload is less effective at longer transfer times (30s and 60s). We observe frequent partial V2V transfers, where the data transfer did not complete within the transfer time. These partial transfers are the primary cause of the reduced ratio. In urban scenarios, it is difficult for vehicles to remain within V2V coverage for a long time. From the results, we can gather the video data from the CLs in a scalable manner with the short transfer time. This implies that reducing the V2V transfer time in VMC is critical.

We also analyze how the CM ratio affects data uploading. The data upload ratio for different CM ratios is shown in Figure 6b. With a 1% CM ratio, the upload ratio is the lowest (3.8%), primarily due to the very small number of CMs—only one or two CMs typically complete V2V transfers in this scenario. When the CM ratio exceeds 25%,



(a) Data transfer ratio of collaborative upload

(b) Data transfer ratio of uploader vehicle

Fig. 6: Collaborative upload performance [22]. The figure illustrates how the different communication technologies (V2V Wi-Fi and V2C) used speed-up the collaborative upload mediated by a VMC.

the upload ratio saturates at approximately 60%. Thus, a 25% CM ratio is sufficient for collaborative uploading in urban areas. This indicates that selecting an appropriate CM ratio is vital for the proposed method. Moreover, we achieve comparable upload ratios with fewer CMs by leveraging data replication [26].

4.2 Collaborative Data Download

One potential use case of a VMC is a collaborative downloading service in which vehicles retrieve data from a remote server and then form VMCs to share the content with nearby vehicles via V2V communican [27]. Such collaboration helps reduce both latency and operational costs associated with large-scale content distribution. In this part, we focus on one example application: distributing HD map updates. The first vehicle that downloads the HD map becomes the VMC leader (CL) and establishes a VMC. Vehicles in proximity then join as VMC members and collaboratively obtain the HD map from the leader. Through field trials with multiple vehicles, the effectiveness of this collaborative HD map downloading service was demonstrated. Using a Wi-Fi–based VMC, outdoor experiments show that VMC-assisted collaboration can significantly reduce the communication cost of HD map updates.

System Design of Collaborative Downloading: Figure 7 illustrates the system overview of collaborative download by VMCs. In the setup, Wi-Fi is used for V2V communication, while LTE is used for V2C communication. When a vehicle enters a target region, it downloads the HD map from a remote server via V2C and becomes a VMC leader once it encounters other connected vehicles. In the implementation, vehicles within a 250-meter radius of the leader are invited to join the VMC over V2V communication. The VMC leader then advertises the availability of the HD map, and nearby vehicles request the map segments from the leader. These VMC members download the HD map using V2V communication.

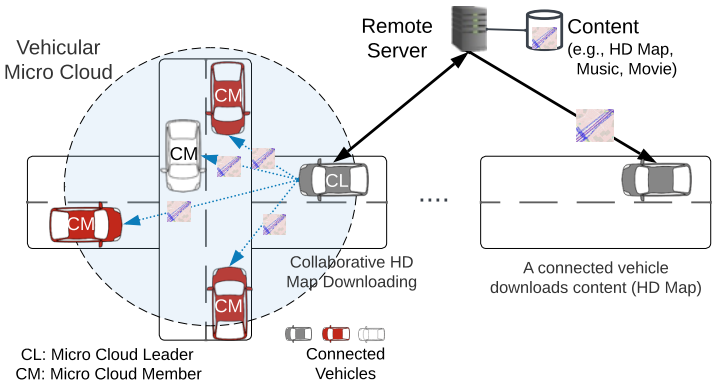
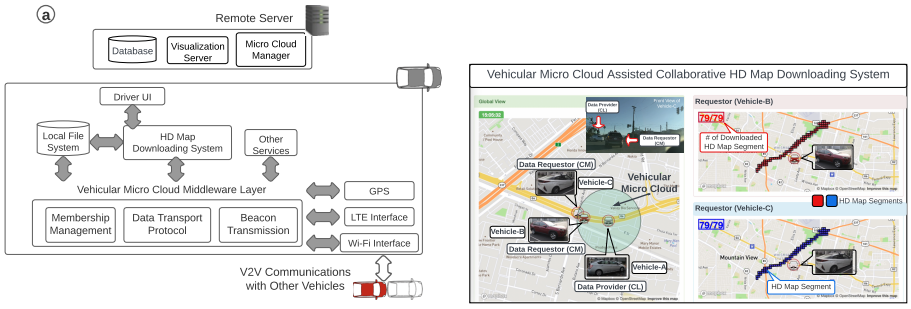


Fig. 7: Collaborative Data Download by VMC [27]. The figure shows how download and further distribution within a VMC facilitates the content download process.

For the experiments, HD maps of California State Route 237 (CA-237) were used, a 4.2km, 4-lane freeway segment (2 lanes per direction) between El Camino Real in Mountain View and Interstate 680 in Milpitas. The map is divided into 100m × 100m tiles, producing 79 files totaling 1.7MB. The VMC leader downloads the HD map data via V2C and then distributes it to nearby vehicles using the VMC-assisted collaborative downloading service.



(a) Architecture of PoC System (b) PoC Demonstration with 3 Test Vehicles

Fig. 8: Collaborative download [27]. The figure shows the general architecture of the collaborative download proof-of-concept (PoC) system as well as a snapshot of the collaborative data dissemination process, where a single vehicle first downloads the HD map from the remote server and subsequently redistributes it to nearby vehicles via V2V communication.

Collaborative Download Implementation: Figure 8 presents the overall architecture of the in-vehicle system and the Proof-of-Concept (PoC) field trials. As shown in

Figure 8a, the system includes a remote server with a database that stores the HD map data. The server also hosts a visualization service where vehicles upload their locations and HD map download status, enabling centralized monitoring of multiple vehicles in real time. The in-vehicle system incorporates a middleware layer that manages VMC operations, including VMC formation, membership management, and role handover when members leave the VMC. It interfaces with other in-vehicle components, such as the local computing system and sensor modules, via the ROS API.[2]

Figure 8b shows the outdoor experiments using three vehicles, labeled Vehicle-A, Vehicle-B, and Vehicle-C. Vehicle-A downloads the HD map from the remote server and becomes the cloud leader, forming the VMC and serving as the data provider. It advertises the availability of the HD map to nearby vehicles. Vehicle-B and Vehicle-C join as VMC members and act as data requesters. Through the VMC, the VMC leader transfers the HD map data to these members, who download the content concurrently.

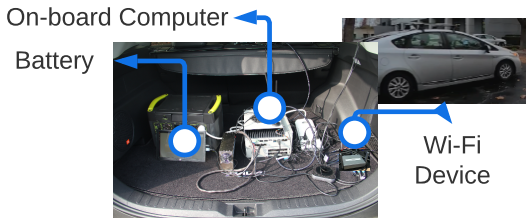


Fig. 9: Vehicle setup [27]. The figure shows photographs of the PoC system with the on-board computer and the Wi-Fi subsystem.

PoC Setup: Figure 9 shows the hardware configuration of the test vehicles. Each vehicle is equipped with an onboard computer that executes the in-vehicle system. Vehicles use USB LTE dongles for V2C communication with the remote server. The onboard computers also include an external Wi-Fi device and a GPS/IMU module (Xsens MTi-100). The VMC leader operates its Wi-Fi interface in Access Point (AP) mode to form the VMC and advertise HD map data. Other vehicles periodically scan for available APs and join the VMC as VMC members when they come within a 250m range of the leader.

Evaluation Results: Figure 10 presents results from one of the PoC experiments. The first vehicle (Vehicle-A) downloads the HD map and encounters another connected vehicle (Vehicle-B) at t = 43s. At this point, Vehicle-A forms a VMC and advertises the available content. Vehicle-B joins as a VMC member and requests the HD map data. The onboarding process for Vehicle-B including VMC discovery and link establishment takes 7.7s. Once the connection is established, the system can transfer the HD map data quickly. However, the link establishment delay remains the primary performance bottleneck.

[2] https://www.ros.org/https://www.ros.org/

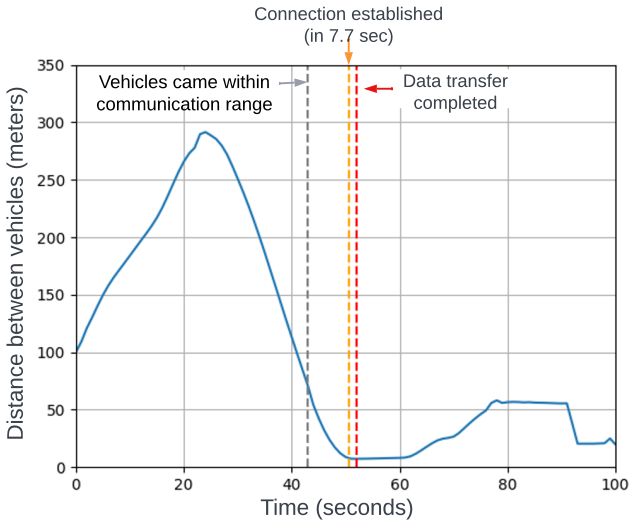


Fig. 10: Result from collaborative downloading PoC experiment [27]. The figure shows the relation between time and distance and the time window for communication between the involved vehicles.

4.3 Summary

In summary, collaborative data up- and downloading demonstrate how VMCs can improve the efficiency of vehicular data exchange. Collaborative uploading enables vehicles to distribute collected data among nearby participants, allowing multiple vehicles to later upload the data through infrastructure networks, which reduces cellular traffic and accelerates large-scale data collection. Similarly, collaborative downloading allows vehicles to share content such as HD map updates through V2V communication, lowering communication costs and reducing download latency. These mechanisms illustrate how VMCs can leverage short-range communication and cooperative vehicle behavior to support scalable data dissemination in connected vehicular environments.

5 Task Offloading in VMCs

This section is based on [19, 34]. We discuss how offloading decisions can be made and how vehicular mobility impacts the execution and completion of distributed tasks. Task offloading is the assignment of computational tasks from user devices to nearby edge servers to reduce latency or improve other performance. In the scope of MEC, task offloading has been extensively studied [4, 35]. This includes work on energy consumption [36], minimizing latency [37], and monetary cost [38], increasing task success rates [39], and resilience [8].

In the dynamic vehicular context, a VMC acts as a virtual edge server [13]. VMC members are in mutual proximity of data storage and computing capacities, enabling task offloading in VMCs to be feasible. Effective task offloading in VMCs prioritizes time-sensitive, low-data-content computational tasks, as delay-tolerant tasks are more efficiently processed in the distant cloud, while high-volume data suffer from extended transmission times that are unsuitable for highly dynamic vehicular networks.

The primary objectives of task offloading in VMCs are: (1) minimizing latency, (2) maximizing the number of tasks completed before their deadlines, (3) minimizing task failure rates due to inaccurate mobility prediction, and (4) achieving a fair workload distribution among vehicles. By leveraging the local computation capabilities of vehicles, task offloading in VMCs can provide lower latency compared to cloud computing [40], enable real-time processing, and do not rely on centralized infrastructure.

5.1 Background

Task offloading research can have many optimization goals, such as reducing energy consumption [36], minimizing latency [37], lowering monetary cost [38], and improving task success rates [39]. Early non-learning-based work jointly optimized offloading decisions and resource allocation. Mao et al. [39] considered system states such as transmission energy and CPU capacity to reduce execution costs and failures, while Hossain et al. [37] leveraged small-cell MEC to lower latency. Tran and Pompili [36] minimized a weighted sum of delay and energy via convex optimization, and Zhao et al. [41] used a distributed algorithm to reduce average delay. Although effective, these methods rely on simplified network assumptions that overlook complex urban mobility.

Learning-based solutions further improve performance. Wu et al. [42] applied DQN for energy minimization, and Qin et al. [43] used multi-armed bandit)(MAB) to reduce delay and enhance resource utilization. However, these models were trained in controlled environments with limited mobility realism, which restricts their applicability in dynamic urban settings. Thus, mobility-aware studies incorporate vehicle movement into offloading decisions. Cha et al. [44] predicted link duration to prevent failures caused by sudden disconnections, inspiring researchers that mobility prediction can effectively reduce offloading failures under certain conditions.

Shuai et al. [45] proposed a deep reinforcement learning (DRL)-based adaptive task offloading strategy for vehicular edge computing, using a single-actor-critic deep-Q-learning (DQL) framework to minimize processing delay, but its single-objective focus makes it less suitable for multi-objective needs in reality. Farimani et al. [46] proposed a deadline-aware task offloading framework based on the DQL algorithm, which effectively incorporates deadlines into reward design but assumes all tasks are offloaded to RSU, overlooking VMC computing potential. Wang et al. [47] explored the application of the twin delayed deep deterministic policy gradient (TD3) algorithm in MEC environments, demonstrating its strong learning capability for task offloading, offering valuable insights despite not involving VMC directly.

In the following, we first present a general system model for task offloading in a VMC followed by selected machine learning based offloading solutions.

5.2 System Model

The VMC system incorporates several elements: vehicles, tasks, and a controller or decentralized agents. A VMC comprises a collection of vehicles in close physical proximity that share their location information, communication, storage, and computing resources to form a distributed edge computing platform. Within a VMC, car members play multiple roles: task generators requiring computing services, and edge computing servers providing diverse computational capabilities, as illustrated in Figure 11 where car colors and generated task colors correspond. In this example, vehicles B and D have spare resources, while A and C are overloaded. Vehicle E has no spare resources nor overloaded, so it processes its own tasks locally. Therefore, A offloads their tasks to B and D and C offloads tasks to D, improving overall time efficiency and promoting a more balanced workload distribution across vehicles.

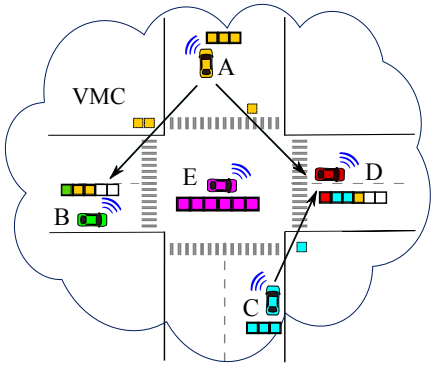


Fig. 11: Task offloading in a VMC [19]. The key idea illustrated is the dual role of vehicles in VMCs: vehicles may generate tasks to be processed and also help other vehicles computing tasks by a given deadline. The arrows indicate how vehicles offload tasks to nearby cars with available capacities, enabling cooperative workload balance and improving the overall task processing efficiency.

Tasks generated by cars are heterogeneous, and three key characteristics significantly impact processing and migration performance: data size, which impacts communication delays; task complexity, which influences the actual processing time on the server; and task deadline, which limits the maximum acceptable time for the sum of communication, waiting, and computation time. Of course, the dwell time of vehicles also matters in the task offloading decision-making optimization problem.

Vehicle Model: Members in a VMC are characterized by three key factors: their dwell time, their computing capabilities, and current resource utilization. To model these, let t and i represent discrete timesteps and car indices, respectively. Each vehicle

Veh_i is associated with either a predicted and upper-bounded dwell period defined by (dwell(i)), or perfect knowledge of dwell time defined by the tuple ($t^{\text{enter}}i, t^{\text{exit}}i$), indicating its availability within the VMC. The total number of vehicles that have ever entered is denoted by $\max i$. In terms of computational ability, vehicle $Vehi_i$ has computing capacities measured in million instructions per second (MIPS), which is directly related to the workload or task complexity unit (million instructions) instead of CPU cycles. We assume that the number of tasks that each vehicle Veh_i generates at each time step t follows a uniform distribution over $[0, n]$.

Task Model: We denote the sequence number of a task by k, where $\max\{k\}$ represents the total number of tasks ever generated within the VMC in a certain time duration. These tasks are also heterogeneous and characterized by three key factors: data size (in megabytes, MB), computational complexity (in millions of instructions, MI), and deadline (in seconds, s). The task metric MI has compatibility with the cars' computing capacity metric (MIPS) and impacts the tasks' execution time, while the MB data size indicates the throughput of transmitted data, deciding the task uplink time together with the channel data rate.

The total delay T_k for task k is the sum of its communication time T_k^{c}, waiting time in the processing car T_k^{w}, and processing time T_k^{p}, formulated as:

$$T_k \triangleq T_k^{\text{w}} + T_k^{\text{c}} + T_k^{\text{p}}.$$

5.3 Centralized Controller vs. Decentralized Agent Models

A central controller in a VMC, such as an elected leader or a RSU allows for global coordination. Of course, when the role is fulfilled by one of the vehicles, it needs to be handed over when this car leaves the VMC. The controller's responsibilities are cyclical at each time slot t:

1. Update the vehicle set by adding new joins (with their location, idleness, and computing capacity) and removing recent departures;
2. If perfect dwell times are assumed, read the dwell time of newly joined vehicles.
3. If not, then predict the remaining time for all current vehicles to inform offloading decisions.
4. Collect information for newly generated tasks, including their data size, computational complexity, and deadlines.
5. Prioritize tasks and calculate potential time consumption and other key factors. Then make optimal offloading decisions based on the task migration protocol and the calculated results.

In decentralized learning-based frameworks, there is no central controller. Instead, each vehicle acts as a learning agent. These agents make task offloading decisions based on observation of the whole environment, i.e., VMC, and its own migration policy. The process for each agent in each timestep t is:

1. Observe the state s_t in the current VMC, e.g., current vehicles' capacities and workloads.

2. Record the attributes of the generated tasks, e.g., data size, deadline, and computational load.
3. Select an offloading action via its actor network.
4. During training, update its network parameters based on rewards.

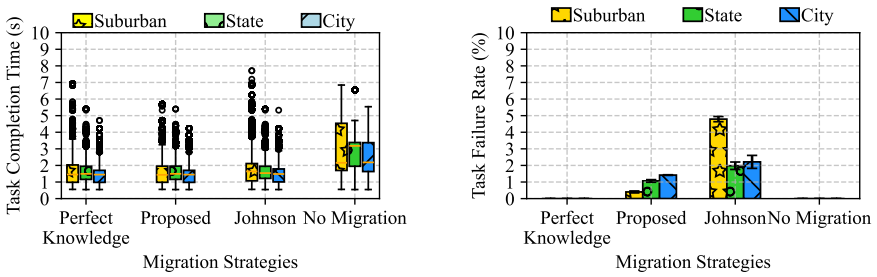
In the following, we introduce two task offloading strategies in detail: a greedy single-objective method [19] that minimizes latency using dwell-time predictions, and a multi-objective learning-based approach [34] that balances time efficiency and vehicle workload fairness using the multi-agent twin delayed deep deterministic policy gradient (MATD3) algorithm.

5.4 Single-Objective Heuristic with Dwell Time Prediction

The algorithm by Zhou et al. [19] is divided into two parts: dwell time prediction and offloading decision making. First, a random forest model is trained to predict upper-bounded vehicle dwell times. This model uses the history of cars' coordinates over a time duration as features, and the duration is determined by the historical maximum task completion time. Then the model is trained, hyperparameter-optimized, and evaluated using standard regression metrics. Second, the central controller has a list of all newly-generated tasks sorted by their deadlines, i.e., following the earliest deadline first (EDF) concept. For each task, it will calculate and filter car candidates that have longer predicted dwell times than the task completion time. The shortest completion time offering car among all candidates will be selected as the task's processor car. If no such candidate exists, then the generator car will process the task locally.

To evaluate the proposed task offloading strategies, simulations were conducted across diverse traffic scenarios. These include a single intersection drawn in SUMO software, three intersections across suburban, city center, and highway in Luxembourg [16], and three intersections across suburban, downtown, and state roads in Nagoya [20]. The last two city scenarios covered different times during a day, such as morning rush hour, afternoon non-peak, and midnight periods [19].

The results (morning rush hour, a downtown intersection in Nagoya) in Figure 12 show that the greedy algorithm with dwell prediction effectively reduces missed deadlines by consistently assigning tasks to the vehicle with the shortest completion time and proper dwell time. Task completion times (cf. Figure 12a) across the three migration strategies are significantly reduced compared to no migration. The proposed model clearly outperforms the Johnson SU dwell prediction approach w.r.t. the task failure rate (due to dwell prediction failure, cf. Figure 12b): The proposed solution exhibits a failure rate of around 0.5–1.5 %, whereas the Johnson SU approach leads to 2.0–5.0 % across all locations in Nagoya. This demonstrates the robustness and effectiveness of this migration mechanism when combined with the dynamic prediction model. Dwell time prediction reduces failures due to mobility, and the greedy algorithm reduces missed deadline cases by always assigning tasks to their specific shortest-time candidate.



(a) Task Completion Time

(b) Task failure rate due to mobility prediction

Fig. 12: Migration performance, Nagoya scenario during morning rush hour [19]. The figure shows the relationship of task completion time and task failure rate in a dynamic environment for different offloading strategies.

5.5 Multi-Objective Learning-Based Approach

In the following, we introduce a MATD3 algorithm, which is to learn optimal task offloading strategies that balance the trade-off between latency and fairness in VMCs [34]. Each vehicle acts as an independent learning agent with a training phase and an execution phase. The actor network of each agent makes continuous decisions of task offloading based on observations, i.e., the state of its VMC. To ensure learning effectiveness and robustness, the primary network has two additional key components: a replay buffer that stores the previous experience and two twin critic networks that work independently to evaluate actions and mitigate overestimation bias. This overestimation bias is a prevailing problem in single-critic DQL. Twin critics eliminate this problem. Furthermore, there is a target network, including one target actor network and two target critic networks, which are slow-updated copies of the actor and critics to compute stable target values and help training converge.

The MATD3 approach is depicted in Figure 13. It dynamically evaluates environmental states to compute the costs and benefits of migration, effectively balancing the trade-off between task completion delay and workload fairness among vehicles. Each agent independently makes offloading decisions via its actor network, while its twin critics learn from the feedback to guide the actor towards optimal policies, and all these are stabilized by target networks.

Selected performance results of the MATD3 algorithm are depicted in Figure 14. Here, the Nagoya morning rush hour scenario [20] (a state road intersection) is used for evaluation. As shown in Figure 14a, the MATD3 approach significantly reduces task completion times during the rush hour, avoids excessive delays. Further information, such as achieving an 80 % on-time completion rate can be found in [34]. While the least-workload method achieves slightly better fairness, the MATD3 maintains a reasonable fairness level and outperforms the other two baselines, which can be seen in [34]. Figure 14b illustrates that MATD3 has a relatively high usage of car computing

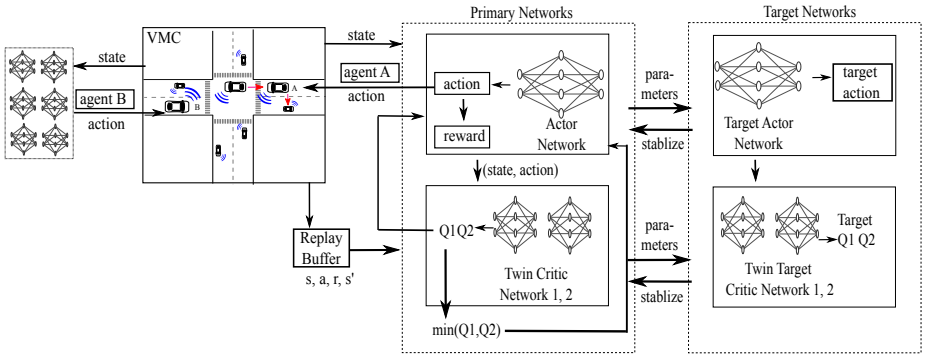
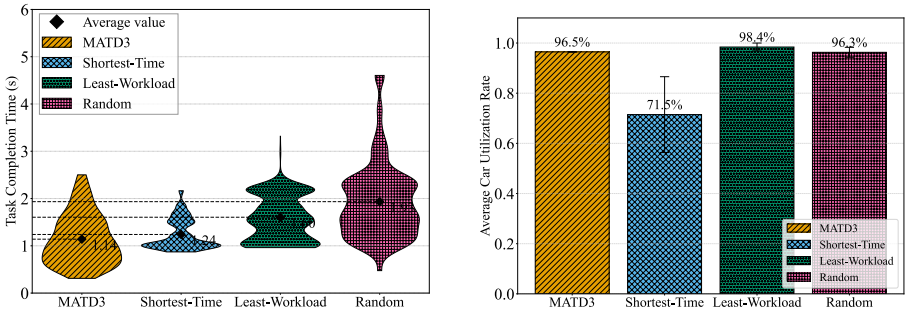


Fig. 13: MATD3 Architecture of the proposed MATD3-based task offloading framework in vehicular edge computing [34]. The figure illustrates the learning process of multiple agents, where each agent uses an actor network for decision making and twin target networks for policy evaluation, while the three corresponding target networks provide stable reference values.

resources, while the shortest-time strategy does not. Comprehensively, experimental results demonstrate that this multi-agent combination significantly enhances comprehensive migration performance, outperforming random, fairness-oriented, and shortest-time-oriented baselines by effectively managing delay-sensitive tasks with deadlines in dynamic vehicular environments.



(a) Task completion time (b) Utilization

Fig. 14: Nagoya scenario: time efficiency and workload fairness performance [34]. Values shown contrast the task completion time vs. average utilization rate for the different approaches to understand the trade-off.

5.6 Summary

Task offloading in VMCs allows computational workloads to be dynamically distributed across vehicles in the same VMC. By leveraging available and idle resources for a temporary short time, low-intensive but high time-efficient tasks can be executed with reduced latency compared to centralized cloud processing; moreover, it can realize better resource allocation compared to fully localized processing.

6 Developing New DRL-based Offloading Concepts

This section is based on [48]. VMCs provide distributed computational resources that can support the collaborative execution of tasks by participating vehicles. However, the highly dynamic nature of vehicular environments makes efficient task offloading and resource allocation challenging. Unlike traditional systems where resource allocation primarily handles dynamic application demands for stable and static resources, VMC environments must deal not only with demand that is dynamic but also with resource availability that is inherently dynamic. Vehicle mobility, fluctuating wireless link quality, and heterogeneous on-board processing capabilities create a computational environment that is both highly non-stationary and heterogeneous. As a result, efficient resource management in VMCs requires mechanisms that can adapt continuously and reliably under uncertainty. To address this challenge, recent researchers explore the use of DRL adaptive scheduling policies that can respond to rapidly variant system conditions.

6.1 VE-SIM – A Simulation Framework for Realistic Evaluation

Realistic simulation plays a crucial role in the evaluation of DRL-based resource management for VMCs. Although numerous studies demonstrate promising results for machine learning (ML)-based approaches, many rely on simplified evaluation setups that neglect realistic mobility patterns, communication variability, among others. These abstractions often lead to optimistic conclusions, causing DRL policies to fail generalizing beyond the scenario in which they were trained. To obtain meaningful insights into system performance, and to ensure that learned policies remain robust under real world conditions, simulation frameworks must accurately capture the dynamism and heterogeneity inherent to VMCs.

To this end, Memedi et al. [48] introduced VE-SIM – a simulation framework specifically designed for ML-based studies in vehicular environments. Figure 15 shows the architecture of VE-SIM. It combines three core components to model VMC environments. *SimPy* serves as the discrete-event simulation engine, representing the temporal evolution of tasks, queues, and resource states. *SUMO* provides microscopic mobility traces, ensuring that vehicle movements influence communication and computation

events at fine granularity. *PyTorch* acts as the machine learning backend, enabling the implementation and training of DRL algorithms.

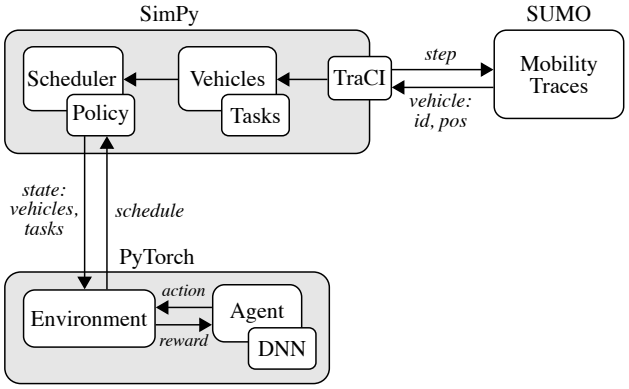


Fig. 15: Main components of VE-SIM simulation framework [48]. The figure shows how the framework integrates SUMO, SimPy, and PyTorch, enabling task generation, real-time traffic mobilities, and DRL training to realize scheduling policies.

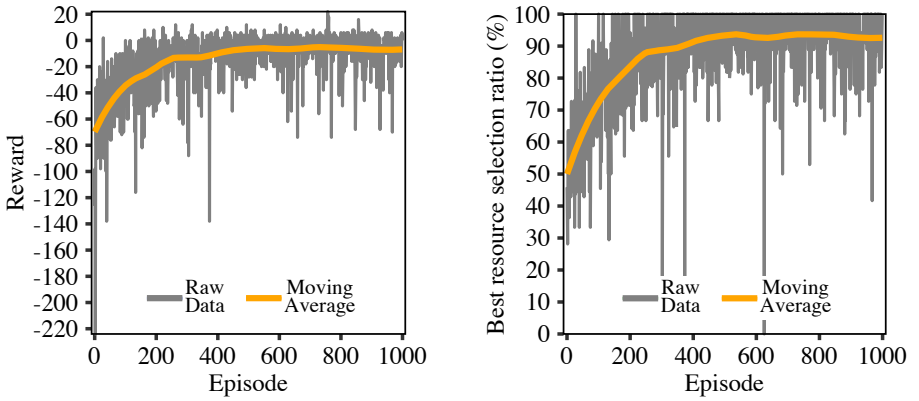
This integrated architecture allows researchers to study how mobility patterns, varying traffic density, and heterogeneous processing capabilities affect both the DRL learning process and the behavior of resulting policies. By capturing these interactions, VE-SIM supports evaluations that more closely mirror real-world VMC setups.

6.2 System Model: Tasks, Vehicles and Scheduler

The system model in VE-SIM reflects the main components of a VMC. Vehicles act as mobile computing resources with heterogeneous processing capacities and time-varying availability. Tasks represent computational jobs defined by complexity, deadlines, and potential priorities. A central scheduler continuously observes the system state and assigns tasks to suitable vehicles according to a chosen policy.

For DRL-based scheduling, the environment provides an observation vector containing task attributes, information about available vehicles. The agent, which implements a DQL algorithm, selects actions by mapping each task to one of the available resources. Rewards favor decisions that assign tasks to vehicles capable of completing them before their deadlines, while penalizing suboptimal matches.

Figure 16 illustrates both the training performance of the DQN agent (shown in Figure 16a as the reward obtained per episode) and its decision quality during inference, represented by the ratio at which the agent selects the vehicle with the highest processing capacity among the available candidates (Figure 16b). Preliminary evaluations demonstrate that the agent rapidly converges to a stable policy after roughly 250 training episodes and increasingly selects the optimal resource as training progresses. This



(a) Collected rewards per episode (gray) and moving average (orange).

(b) Ratio of selecting the best resource in an episode's action space (gray) and moving average (orange).

Fig. 16: Learning performance of the RL agent [48]. The figure illustrates the speed of convergence in terms of reward and the ability to select the best matching resource.

behavior persists even under fluctuating task arrival rates and varying computational loads, indicating that the framework effectively supports DRL training under dynamic conditions. In general, these findings highlight the potential of DRL-based schedulers to achieve robust and efficient task allocation in VMCs, despite inherent variability and unpredictability of the environment.

6.3 Summary

In summary, the dynamic and heterogeneous nature of VMCs requires adaptive resource management strategies that are able to deal with rapidly changing systems. The VE-SIM framework provides a simulation environment that is close-to-realistic and integrates real-time traffic mobility, wireless communication, and real-time DRL-based task scheduling and mapping approaches. First results demonstrate that DRL agents can effectively learn task allocation strategies, even under fluctuating task and computing conditions.

7 Challenges and Practical Considerations

7.1 Industry Perspective and First Pilots

The transition from conceptual VMC architectures toward practical deployments is increasingly supported by both early pilot demonstrations and ongoing industry standardization activities. These efforts collectively emphasize distributed intelligence, opportunistic connectivity, and localized cooperation among vehicles. In particular, the IEEE 802.11 Automotive Topic Interest Group (Auto TIG) and the Wi-Fi Alliance automotive initiatives highlight the importance of WLAN-based connectivity as a complementary technology for connected vehicles.

The IEEE 802.11 Automotive Topic Interest Group was established to address how WLAN stations can seamlessly connect to external access points in high-mobility vehicular environments. The effort specifically targets challenges such as fast association, authentication, and efficient data offloading when vehicles move across heterogeneous connectivity domains [49]. The Automotive TIG further investigates automotive use cases, including HD-map downloads, sensor-data sharing, and multi-vehicle coordination, while identifying gaps in existing WLAN standards related to roaming, mobility support, and service discovery [50]. These requirements align closely with VMC concepts, where vehicles opportunistically form groups and exchange data locally to reduce latency and improve throughput.

Complementing standardization, industry organizations such as the Wi-Fi Alliance are actively promoting Wi-Fi connectivity for automotive applications. The Wi-Fi Alliance emphasizes secure and automatic connection to trusted roadside networks, interoperability with 5G and C-V2X technologies, and new service opportunities enabled by edge analytics and roaming frameworks [51]. Such initiatives highlight the role of Wi-Fi as part of a heterogeneous connectivity fabric supporting distributed vehicular intelligence and collaborative applications.

Early pilot demonstrations further validate VMC concepts in real-world scenarios. A first group of demonstrations focuses on collaborative safety, where micro cloud members share observations to detect and mitigate risky driving behavior. For example, the connected unsafe driving detection pilot demonstrates distributed detection of hazardous maneuvers and collaborative mitigation strategies enabled by VMCs [52]. Similarly, collaborative response to unsafe maneuvers illustrates how approaching vehicles form VMC and coordinate to prevent collisions caused by aggressive turns or unexpected movements [53].

A second group of demonstrations addresses collaborative data storage and situational awareness. Vehicular collaborative data storage for road incident monitoring allows vehicles near an event to form VMC and locally store and propagate incident information, improving awareness for approaching vehicles [54]. Additional work on collaborative data storage using VMCs shows that distributed replication mechanisms can maintain data availability despite dynamic vehicle mobility [55].

A third group explores collaborative data distribution and decision support by VMCs. VMC assisted HD-map downloading demonstrates collaborative map distribution among vehicles, reducing infrastructure load [23]. Similarly, intelligent intersec-

tion management [56] and lane change advisory systems [57] use VMCs to coordinate decision-making and reduce traffic conflicts.

More recent demonstrations extend these VMCs toward collaborative data uploading, highlighting the bidirectional nature of edge–cloud interaction [22]. In particular, the Los Angeles case study on collaborative uploading demonstrates that vehicles can first exchange data via V2V links within a micro cloud and then opportunistically upload aggregated data to the cloud when connectivity becomes available. This approach reduces redundant cellular transmissions and improves scalability, especially in dense urban environments, further reinforcing the role of vehicular micro clouds as intermediate edge aggregation layers.

Table 4: Early Pilot Demonstrations of VMC Applications

Category	**Demonstration**	**Functionality**	**Contribution**
Collaborative Safety	Connected Unsafe Driving Detection and Management	Collaborative risk detection	Hazard mitigation
Collaborative Safety	Responding to Unsafe Maneuver of Approaching Vehicles	Coordinated maneuver response	Collision avoidance
Collaborative Safety	Vehicular Collaborative Data Storage for Road Incident Monitoring	Local incident sharing	Situational awareness
Collaborative Storage	Collaborative Data Storage by Vehicular Micro Cloud	Distributed replication	Edge data persistence
Collaborative Download	VMC Assisted HD Map Downloading	Collaborative data distribution	Reduced infrastructure load
Collaborative Upload	Collaborative Data Uploading (Los Angeles Case Study)	V2V-assisted aggregation	Reduced cellular overhead
Collaborative Coordination	VMC Intelligent Intersection Management	Collaborative control	Safer intersections
Advisory Systems	Lane Change Advisory by Vehicular Micro Clouds	Decision support	Safer lane changes

Together, these pilot deployments demonstrate that VMCs can support safety-critical applications, distributed storage, and collaborative decision-making. Importantly, the functionality explored in these demonstrations aligns with the requirements identified by IEEE 802.11 automotive standardization efforts, such as multi-vehicle coordination, fast connectivity establishment, and localized data exchange. At the same time, the Wi-Fi Alliance automotive initiative highlights the feasibility of integrating Wi-Fi connectivity into a heterogeneous vehicular communication ecosystem. These combined efforts indicate that VMCs represent a practical building block for next-generation collaborative intelligent transportation systems, bridging research prototypes and emerging industry deployment pathways.

Table 5: Pilot Demonstrations by Architectural Role

Architectural Role	**Example Pilots**	**Supported Services**
Edge Safety Intelligence	Unsafe driving detection, maneuver response	Risk prediction and avoidance
Distributed Data Layer	Collaborative storage, incident monitoring	Local data persistence
Collaborative Data Distribution	HD map collaborative download	Efficient content sharing
Collaborative Data Upload	LA collaborative uploading	Scalable aggregation
Collaborative Traffic Management	Intersection management, lane advisory	Coordinated decisions

7.2 Further Challenges

In the scope of this chapter, we discussed the advantages and opportunities of vehicular micro clouds to serve as virtualized edge servers in dynamic vehicular environments. While the concepts and strategies presented demonstrate promising results, several critical challenges remain that need to be addressed in order to realize real-world adoption.

Security Concerns: First, all data exchange must be protected from unwanted modification, overhearing, replay, and other types of security attacks. Ensuring secure communication, data integrity, and protection against malicious nodes is a must to protect the system, the users, and enable resilient operation of data communication and distributed computing. Although current industrial pilot deployments, such as collaborative risk detection and coordinated maneuver response in open vehicular environments, have addressed several safety-related challenges in vehicular networks, important gaps still remain or more security concerns will appear with the development of more brand-new technologies.

Privacy Issues: Second, data management and task offloading can involve transmitting potentially private data, such as vehicle sensor readings, camera feeds, or location traces, to other vehicles that pose critical privacy issues. Appropriate solutions for privacy preserving data sharing need to be integrated into the VMC concept. This also holds for the identities of the participating vehicles.

Scalability: Third, scalability represents a major challenge, particularly in dense traffic environments. If many vehicles are joining and leaving VMCs, this churn can lead to significant control overhead and resource management complexity. Efficient cluster formation and membership management mechanisms are required to maintain system stability under high mobility. This issue is particularly critical in real-world industrial deployments, where high vehicle mobility and dynamic network conditions can significantly increase coordination overhead.

Interoperability: Fourth, interoperability is a critical issue considering large-scale deployment. Vehicles from different car makers may use different hardware platforms, software stacks, and even communication protocols. Standardized interfaces will be necessary to allow seamless cooperation across the heterogeneous vehicular ecosystem. Industry standardization efforts, such as multi-vehicle coordination, fast connectivity establishment, and localized data exchange, together highlight the importance of stan-

dards and protocols to enable seamless cooperation across heterogeneous vehicular systems.

8 Conclusion

Vehicular micro clouds (VMCs) represent a powerful and flexible realization of distributed computing within the edge–cloud continuum, specifically tailored to the highly dynamic nature of vehicular environments. By enabling nearby vehicles to cooperatively pool computation, storage, and communication resources, VMCs bridge the gap between purely local processing and latency-intensive cloud-based solutions. This chapter has reviewed the fundamental concepts, architectures, and mechanisms that underpin VMCs, highlighting their role as virtual edge servers capable of supporting a wide range of collaborative vehicular applications.

We first introduced the system model and architectural principles of VMCs, including their formation, management, and integration into macro–micro–cloud hierarchies. Particular attention was given to the challenges arising from vehicular mobility, such as short dwell times, frequent topology changes, and volatile resource availability. To address these challenges, the chapter surveyed key mechanisms for data retention and recovery, demonstrating how mobility-aware protocols and adaptive communication strategies can significantly improve data availability despite extreme node churn. Furthermore, the use of coded caching was discussed as an effective means to reduce channel load and accelerate data access, illustrating how information-theoretic techniques can be successfully adapted to highly dynamic vehicular networks.

Beyond storage, the chapter examined collaborative data up- and downloading as well as task offloading in VMCs. These use cases showcase how VMCs can reduce reliance on cellular infrastructure, lower operational costs, and enable low-latency processing of time-critical tasks. In particular, learning-based task offloading strategies were reviewed, ranging from heuristic dwell-time-aware approaches to multi-agent reinforcement learning methods that jointly optimize latency, reliability, and workload fairness. Together, these techniques demonstrate that VMCs can serve as a viable platform for distributed intelligence directly embedded in vehicular systems. Looking forward, VMCs are expected to play an increasingly important role in future intelligent transportation systems, especially in the context of 5G/6G networks and large-scale deployment of AI-driven vehicular applications. Open research challenges remain in scalable coordination, robust learning under uncertainty, security and trust management, and seamless integration with fixed edge and cloud infrastructures. Addressing these challenges will be essential for realizing the full potential of VMCs as a cornerstone of resilient, efficient, and intelligent vehicular computing environments.

References

[1] Christoph Sommer and Falko Dressler. *Vehicular Networking*. Cambridge University Press, 2014. doi: 10.1017/CBO9781107110649.

[2] Georgios Karagiannis, Onur Altintas, Eylem Ekici, Geert Heijenk, Boangoat Jarupan, Kenneth Lin, and Timothy Weil. Vehicular Networking: A Survey and Tutorial on Requirements, Architectures, Challenges, Standards and Solutions. *IEEE Communications Surveys & Tutorials*, 13(4):584–616, November 2011. doi: 10.1109/SURV.2011.061411.00019.

[3] Kan Zheng, Qiang Zheng, Periklis Chatzimisios, Wei Xiang, and Yiqing Zhou. Heterogeneous Vehicular Networking: A Survey on Architecture, Challenges, and Solutions. *IEEE Communications Surveys & Tutorials*, 17(4):2377–2396, June 2015. doi: 10.1109/COMST.2015.2440103.

[4] Pavel Mach and Zdenek Becvar. Mobile Edge Computing: A Survey on Architecture and Computation Offloading. *IEEE Communications Surveys & Tutorials*, 19(3):1628–1656, March 2017. doi: 10.1109/COMST.2017.2682318.

[5] Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled Ben Letaief. A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Communications Surveys & Tutorials*, 19(4):2322–2358, October 2017. doi: 10.1109/comst.2017.2745201.

[6] Claudio Ettore Casetti, Carla Fabiana Chiasserini, Falko Dressler, Agon Memedi, Diego Gasco, and Elad Michael Schiller. AI/ML-based Services and Applications for 6G-Connected and Autonomous Vehicles. *Elsevier Computer Networks*, 255: 110854, December 2024. doi: 10.1016/j.comnet.2024.110854.

[7] L. Baresi, D. F. Mendonça, M. Garriga, S. Guinea, and G. Quattrocchi. A Unified Model for the Mobile-Edge-Cloud Continuum. *ACM Transactions on Internet Technology*, 19(2):1–21, April 2019. doi: 10.1145/3226644.

[8] Doğanalp Ergenç, Agon Memedi, Mathias Fischer, and Falko Dressler. Resilience in Edge Computing: Challenges and Concepts. *Foundations and Trends® in Networking*, 14(4):254–340, May 2025. doi: 10.1561/1300000074.

[9] Mario Gerla. Vehicular Cloud Computing. In *11th IFIP/IEEE Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net 2012)*, pages 152–155, Ayia Napa, Cyprus, June 2012. IEEE. doi: 10.1109/MedHocNet.2012.6257116.

[10] Falko Dressler, Philipp Handle, and Christoph Sommer. Towards a Vehicular Cloud - Using Parked Vehicles as a Temporary Network and Storage Infrastructure. In *15th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2014), ACM International Workshop on Wireless and Mobile Technologies for Smart Cities (WiMobCity 2014)*, pages 11–18, Philadelphia, PA, August 2014. ACM. doi: 10.1145/2633661.2633671.

[11] Falko Dressler, Carla Fabiana Chiasserini, Frank H. P. Fitzek, Holger Karl, Renato Lo Cigno, Antonio Capone, Claudio Ettore Casetti, Francesco Malandrino, Vincenzo Mancuso, Florian Klingler, and Gianluca Antonio Rizzo. V-Edge: Virtual Edge Computing as an Enabler for Novel Microservices and Cooperative Computing. *IEEE Network*, 36(3):24–31, May 2022. doi: 10.1109/MNET.001.2100491.

[12] Takamasa Higuchi, Joshua Joy, Falko Dressler, Mario Gerla, and Onur Altintas. On the Feasibility of Vehicular Micro Clouds. In *9th IEEE Vehicular Networking Conference (VNC 2017)*, pages 179–182, Turin, Italy, November 2017. IEEE. doi: 10.1109/VNC.2017.8275621.

[13] Falko Dressler, Gurjashan Singh Pannu, Florian Hagenauer, Mario Gerla, Takamasa Higuchi, and Onur Altintas. Virtual Edge Computing Using Vehicular Micro Clouds. In *IEEE International Conference on Computing, Networking and Communications (ICNC 2019)*, Honolulu, HI, February 2019. IEEE. doi: 10.1109/ICCNC.2019.8685481.

[14] Florian Hagenauer, Christoph Sommer, Takamasa Higuchi, Onur Altintas, and Falko Dressler. Vehicular Micro Clouds as Virtual Edge Servers for Efficient Data Collection. In *23rd ACM International Conference on Mobile Computing and Networking (MobiCom 2017), 2nd ACM International Workshop on Smart, Autonomous, and Connected Vehicular Systems and Services (CarSys 2017)*, pages 31–35, Snowbird, UT, October 2017. ACM. doi: 10.1145/3131944.3133937.

[15] David Eckhoff, Christoph Sommer, Reinhard German, and Falko Dressler. Cooperative Awareness At Low Vehicle Densities: How Parked Cars Can Help See Through Buildings. In *IEEE Global Communications Conference (GLOBECOM 2011)*, Houston, TX, December 2011. IEEE. doi: 10.1109/GLOCOM.2011.6134402.

[16] Lara Codeca, Raphaël Frank, and Thomas Engel. Luxembourg SUMO Traffic (LuST) Scenario: 24 Hours of Mobility for Vehicular Networking Research. In *7th IEEE Vehicular Networking Conference (VNC 2015)*, Kyoto, Japan, December 2015. IEEE. doi: 10.1109/VNC.2015.7385539.

[17] Gurjashan Singh Pannu, Seyhan Ucar, Takamasa Higuchi, Onur Altintas, and Falko Dressler. Dwell Time Estimation at Intersections for Improved Vehicular Micro Cloud Operations. *Elsevier Ad Hoc Networks*, 122:102606, November 2021. doi: 10.1016/j.adhoc.2021.102606.

[18] Max Schettler, Gurjashan Singh Pannu, Seyhan Ucar, Takamasa Higuchi, Onur Altintas, and Falko Dressler. Learning-based Dwell Time Prediction for Vehicular Micro Clouds. In *18th IEEE International Conference on Mobility, Sensing and Networking (MSN 2022)*, pages 542–549, Guangzhou, China, December 2022. IEEE. doi: 10.1109/MSN57253.2022.00091.

[19] Ziqi Zhou, Agon Memedi, Chunghan Lee, Seyhan Ucar, Onur Altintas, and Falko Dressler. Task Migration with Deadlines using Machine Learning-based Dwell Time Prediction in Vehicular Micro Clouds. *Elsevier High-Confidence Computing*, 5(2):100314, June 2025. doi: 10.1016/j.hcc.2025.100314.

[20] Takamasa Higuchi, Lei Zhong, and Ryokichi Onishi. NUMo: Nagoya Urban Mobility Scenario for City-Scale V2X Simulations. In *15th IEEE Vehicular Networking Conference (VNC 2024)*, pages 17–24, Kobe, Japan, May 2024. IEEE. doi: 10.1109/VNC61989.2024.10575975.

[21] Gurjashan Singh Pannu, Florian Hagenauer, Takamasa Higuchi, Onur Altintas, and Falko Dressler. Keeping Data Alive: Communication Across Vehicular Micro Clouds. In *20th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2019)*, Washington, D.C., June 2019. IEEE. doi: 10.1109/WoWMoM.2019.8792973.

[22] Chunghan Lee, Takamasa Higuchi, Seyhan Ucar, Naoya Kaneko, Onur Altintas, and Kentaro Oguchi. Is Collaborative Data Uploading Feasible? A Case for Los Angeles with Vehicular Micro Clouds. In *100th IEEE Vehicular Technology Conference (VTC2024-Fall)*, Washington, D.C., October 2024. IEEE. doi: 10.1109/vtc2024-fall63153.2024.10757687.

[23] Seyhan Ucar, Takamasa Higuchi, and Onur Altintas. Demo: Vehicular Micro Cloud Assisted Collaborative High-Definition Map Downloading. In *14th IEEE Vehicular Networking Conference (VNC 2023), Demo Session*, pages 177–178, Istanbul, Turkey, April 2023. IEEE. doi: 10.1109/VNC57357.2023.10136280.

[24] Gurjashan Singh Pannu, Seyhan Ucar, Takamasa Higuchi, Onur Altintas, and Falko Dressler. Data Sharing in Virtual Edge Computing using Coded Caching. In *14th IEEE Vehicular Networking Conference (VNC 2023)*, pages 104–111, Istanbul, Turkey, April 2023. IEEE. doi: 10.1109/VNC57357.2023.10136281.

[25] Urs Niesen and Mohammad Ali Maddah-Ali. Coded Caching With Nonuniform Demands. *IEEE Transactions on Information Theory*, 63(2):1146–1158, February 2017. doi: 10.1109/tit.2016.2639522.

[26] Chunghan Lee, Obasi Mba, Seyhan Ucar, Emrah Akin Sisbot, Yashar Farid, Onur Altintas, and Kentaro Oguchi. Replica-Based Collaborative Data Upload by Vehicular Micro Clouds. In *16th IEEE Vehicular Networking Conference (VNC 2025), Poster Session*, Porto, Portugal, June 2025. IEEE.

[27] Takamasa Higuchi, R. Vince Rabsatt, Mario Gerla, Onur Altintas, and Falko Dressler. Cooperative Downloading in Vehicular Heterogeneous Networks at the Edge. In *IEEE Global Communications Conference (GLOBECOM 2019), Workshop on New and Disruptive Technologies and Applications for Mobile Edge/Fog Computing (MobileEdgeCom 2019)*, Waikoloa, HI, December 2019. IEEE. doi: 10.1109/GCWkshps45667.2019.9024655.

[28] AECC. General Principle and Vision. White Paper 2.1.0, Automotive Edge Computing Consortium, December 2018. URL https://aecc.org/wp-content/uploads/2019/04/AECC_White_Paper_v2.1_003.pdf.

[29] Rui Meireles, Antonio Rodrigues, Andrei Stanciu, Ana Aguiar, and Peter Steenkiste. Exploring Wi-Fi Network Diversity for Vehicle-To-Infrastructure Communication. In *12th IEEE Vehicular Networking Conference (VNC 2020)*, Virtual Conference, December 2020. IEEE. doi: 10.1109/vnc51378.2020.9318407.

[30] Furong Yang, Andrea Ferlini, Davide Aguiari, Davide Pesavento, Rita Tse, Suman Banerjee, Gaogang Xie, and Giovanni Pau. Revisiting WiFi offloading in the wild for V2I applications. *Elsevier Computer Networks*, 202:108634, January 2022. doi: 10.1016/j.comnet.2021.108634.

[31] Johannes Bach, Jacob Langner, Stefan Otten, Marc Holzapfel, and Eric Sax. Data-driven development, a complementing approach for automotive systems engineering. In *IEEE International Systems Engineering Symposium (ISSE 2017)*, Vienna, Austria, October 2017. IEEE. doi: 10.1109/syseng.2017.8088295.

[32] Mathias Johanson, Stanislav Belenki, Jonas Jalminger, Magnus Fant, and Mats Gjertz. Big Automotive Data: Leveraging large volumes of data for knowledge-driven product development. In *IEEE International Conference on Big Data (Big Data 2014)*, pages 736–741, Washington, D.C., October 2014. IEEE. doi: 10.1109/bigdata.2014.7004298.

[33] Hao Yang, Yashar Zeiynali Farid, and Kentaro Oguchi. Cost and Benefit Analysis of An Integrated Traffic Management System Utilizing Connected Vehicles. In *IEEE International Conference on Networking, Sensing and Control (ICNSC 2022)*, Shanghai, China, December 2022. IEEE. doi: 10.1109/icnsc55942.2022.10004064.

[34] Ziqi Zhou, Agon Memedi, Chunghan Lee, Seyhan Ucar, Onur Altintas, and Falko Dressler. Fairness-Aware Multi-Agent Learning-based Task Offloading in Dynamic Vehicular Scenarios. In *IEEE International Conference on Metaverse Computing, Networking and Applications (MetaCom 2025)*, pages 261–269, Seoul, South Korea, August 2025. IEEE. doi: 10.1109/MetaCom65502.2025.00049.

[35] Kai Li, Yingping Cui, Weicai Li, Tiejun Lv, Xin Yuan, Shenghong Li, Wei Ni, Meryem Simsek, and Falko Dressler. When Internet of Things meets Metaverse: Convergence of Physical and Cyber Worlds. *IEEE Internet of Things Journal*, 10 (5):4148–4173, March 2023. doi: 10.1109/JIOT.2022.3232845.

[36] Tuyen X. Tran and Dario Pompili. Joint Task Offloading and Resource Allocation for Multi-Server Mobile-Edge Computing Networks. *IEEE Transactions on Vehicular Technology*, 68(1):856–868, January 2019. doi: 10.1109/TVT.2018.2881191.

[37] Md Delowar Hossain, Luan N. T. Huynh, Tangina Sultana, Tri D.T. Nguyen, Jae Ho Park, Choong Seon Hong, and Eui-Nam Huh. Collaborative Task Offloading for Overloaded Mobile Edge Computing in Small-Cell Networks. In *34th International Conference on Information Networking (ICOIN 2020)*, pages 717–722, Barcelona, Spain, January 2020. IEEE. doi: 10.1109/ICOIN48656.2020.9016452.

[38] Siyao Cheng, Tian Ren, Hao Zhang, Jiayan Huang, and Jie Liu. A Stackelberg-Game-Based Framework for Edge Pricing and Resource Allocation in Mobile Edge Computing. *IEEE Internet of Things Journal*, 11(11):20514–20530, June 2024. doi: 10.1109/JIOT.2024.3372016.

[39] Yuyi Mao, Jun Zhang, and Khaled B. Letaief. Dynamic Computation Offloading for Mobile-Edge Computing with Energy Harvesting Devices. *IEEE Journal on Selected Areas in Communications*, 34(12):3590–3605, December 2016. doi: 10.1109/JSAC.2016.2611964.

[40] Ejaz Ahmed, Abdullah Gani, Muhammad Khurram Khan, Rajkumar Buyya, and Samee U. Khan. Seamless application execution in mobile cloud computing: Motivation, taxonomy, and open challenges. *Elsevier Journal of Network and Computer Applications*, 52:154–172, June 2015. doi: 10.1016/j.jnca.2015.03.001.

[41] Haitao Zhao, Qixing Zhu, Yue Chen, and Yinyang Zhu. A Research of Task-Offloading Algorithm for Distributed Vehicles. In *IEEE International Conference on Communications (ICC 2020), ICC Workshops 2020 (Workshops)*, pages 1–5, Virtual Conference, June 2020. doi: 10.1109/ICCWorkshops49005.2020.9145331.

[42] Chenhao Wu, Zhongwei Huang, and Yuntao Zou. Delay Constrained Hybrid Task Offloading of Internet of Vehicle: A Deep Reinforcement Learning Method. *IEEE Access*, 10:102778–102788, September 2022. doi: 10.1109/ACCESS.2022.3206359.

[43] Hao Qin, Guoping Tan, Siyuan Zhou, and Yong Ren. Adaptive Learning-Based Multi-Vehicle Task Offloading. In *IEEE/CIC International Conference on Com-*

munications in China (ICCC 2020), pages 1033–1038, Chongqing, China, August 2020. IEEE. doi: 10.1109/ICCC49849.2020.9238793.

[44] Narisu Cha, Celimuge Wu, Tsutomu Yoshinaga, Yusheng Ji, and Kok-Lim Alvin Yau. Virtual Edge: Exploring Computation Offloading in Collaborative Vehicular Edge Computing. *IEEE Access*, 9:37739–37751, January 2021. doi: 10.1109/access.2021.3063246.

[45] Renhao Shuai, Leiyu Wang, Shuaishuai Guo, and Haixia Zhang. Adaptive Task Offloading in Vehicular Edge Computing Networks Based on Deep Reinforcement Learning. In *IEEE/CIC International Conference on Communications in China (ICCC 2021)*, pages 260–265, Xiamen, China, July 2021. IEEE. doi: 10.1109/ICCC52777.2021.9580313.

[46] Mina Khoshbazm Farimani, Soroush Karimian-Aliabadi, Reza Entezari-Maleki, Bernhard Egger, and Leonel Sousa. Deadline-aware task offloading in vehicular networks using deep reinforcement learning. *Expert Systems with Applications*, 249:123622, 2024. doi: 10.1016/j.eswa.2024.123622.

[47] Shuai Wang, Bo Yu, Ning Wang, and Wei Wang. Research on TD3-Based Offloading Strategies for Complex Tasks in MEC Systems. In *IEEE/CIC International Conference on Communications in China (ICCC 2024)*, pages 194–201, Hangzhou, China, August 2024. IEEE. doi: 10.1109/ICCC62609.2024.10941833.

[48] Agon Memedi, Chunghan Lee, Seyhan Ucar, Onur Altintas, and Falko Dressler. Simulator for Reinforcement Learning-based Resource Management in Vehicular Edge Computing. In *16th IEEE Vehicular Networking Conference (VNC 2025), Poster Session*, Porto, Portugal, June 2025. IEEE. doi: 10.1109/VNC64509.2025.11054184.

[49] Status of IEEE 802.11 Automotive TIG. https://www.ieee802.org/11/Reports/auto_update.htm, .

[50] IEEE 802.11 Automotive TIG Technical Report Draft. https://mentor.ieee.org/802.11/dcn/26/11-26-0187-04-auto-draft-final-technical-report.docx, .

[51] Wi-Fi Opportunities for Connected Vehicles. https://wballiance.com/wi-fi-opportunities-for-connected-vehicles.

[52] Seyhan Ucar, Hangquan Zhao, Emrah Akin Sisbot, and Kentaro Oguchi. Demo: Connected Unsafe Driving Detection and Management. In *16th IEEE Vehicular Networking Conference (VNC 2025), Demo Session*, pages 1–2, Porto, Portugal, June 2025. IEEE. doi: 10.1109/VNC64509.2025.11054204.

[53] Seyhan Ucar, Obasi Mba, Chunghan Lee, Emrah Akin Sisbot, and Kentaro Oguchi. Demo: Responding to an Unsafe Maneuver of Approaching Vehicles. In *15th IEEE Vehicular Networking Conference (VNC 2024), Demo Session*, pages 279–280, Kobe, Japan, May 2024. IEEE. doi: 10.1109/VNC61989.2024.10575955.

[54] Seyhan Ucar, Takamasa Higuchi, Chang-Heng Wang, and Onur Altintas. Demo: Vehicular Collaborative Data Storage for Road Incident Monitoring. In *13th IEEE Vehicular Networking Conference (VNC 2021), Demo Session*, pages 123–124, Virtual Conference, November 2021. doi: 10.1109/VNC52810.2021.9644661.

[55] Seyhan Ucar, Takamasa Higuchi, and Onur Altintas. Demo: Collaborative Data Storage by a Vehicular Micro Cloud. In *11th IEEE Vehicular Networking Confer-*

ence (VNC 2019), Demo Session, pages 1–2, Los Angeles, CA, December 2019. IEEE. doi: 10.1109/VNC48660.2019.9062818.

[56] Takamasa Higuchi, Seyhan Ucar, Chang-Heng Wang, and Onur Altintas. Vehicular Micro Cloud as an Enabler of Intelligent Intersection Management. In *12th IEEE Vehicular Networking Conference (VNC 2020), Demo Session*, Virtual Conference, December 2020. IEEE. doi: 10.1109/VNC51378.2020.9318400.

[57] Seyhan Ucar, Takamasa Higuchi, and Onur Altintas. Lane Change Advisory by Vehicular Micro Clouds. In *16th IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS 2019), MASS Workshops 2019 (Worksops)*, pages 164–165, Monterey Park, CA, November 2019. IEEE. doi: 10.1109/MASSW.2019.00041.