Efficient Data Handling in Vehicular Micro Clouds

Florian Hagenauer[a], Takamasa Higuchi[b], Onur Altintas[b], Falko Dressler[*,a]

[a] *Heinz Nixdorf Institute and Dept. of Computer Science, University of Paderborn, Germany*
[b] *Toyota InfoTechnology Center, U.S.A., Mountain View, CA*

Abstract

Wireless communication capabilities currently transform the automotive landscape. Short-range communication technologies enable a wide range of Information and Communication Technology (ICT) application for cars, drivers, and even large-scale Internet of Things (IoT) applications. Many of such applications have complex requirements in particular related to locality of data. Recently, the concept of the *vehicular cloud* has been proposed to address these issues, similar to what is currently investigated in the scope of Mobile Edge Computing (MEC). Forming what we call micro clouds of cars, we establish a virtual roadside infrastructure that can not only support other cars but also complex IoT applications. In this paper, we focus on data management in such micro clouds, i.e., clusters of cars organized in a hierarchical manner. Our micro clouds can provide services in their vicinity and together form macro clouds enabling more complex services and spanning entire cities. We first present an algorithm to form micro clouds at a specific geographic location using a map-based approach. Then, we develop data management services for such dynamic clusters. Concentrating on two services, namely *collect data* for collecting sensor data from vehicles within the micro cloud and forwarding these (possibly in aggregated form) to the macro cloud, and *preserve data* for keeping location-based data at the specified geo-location by continuously handing data from cars leaving to such joining the cluster. Our evaluation results clearly demonstrate the effectiveness of our approach including all the enhancements described in the paper.

Key words: Vehicular Cloud, Micro Cloud, Clustering, Virtual Roadside Infrastructure

1. Introduction

The automotive landscape currently transforms from simple transportation into a complex Information and Communication Technology (ICT) system [15]. One of the main reasons for this transformation are the wireless communication capabilities modern cars come equipped with. These can be short-range technologies like Dedicated Short-Range Communication (DSRC), Wireless LAN (WLAN), or cellular technologies like LTE (or in the near future 5G). The most prominent use-case for such capabilities in vehicles are safety applications. For example, every new car sold in the European Union since April 2018 has to support the *eCall* system. If an emergency occurs, *eCall* transmits, among other information, the vehicles location, the time, and the direction of travel to emergency services via a cellular connection [18]. Due to this timely information, the response time in case of an accident can be significantly reduced. When it comes to applications using short-range communication, the goal is to warn/inform other drivers in the surroundings. Such capabilities are for example part of the ETSI ITS-G5 standard in the form of Cooperative Awareness Messages (CAM) [17].

Beside safety applications, other Intelligent Transportation System (ITS) applications evolve focusing on traffic efficiency and infotainment [45]. Examples include driving route assistance, weather forecasts based on sensor information, or even multimedia streaming. While safety applications usually only require the exchange of short messages within a small geographical range, other applications will often rely on larger amounts of data for performing more complex tasks or transmitting images or videos. Thus, different communication and data management solutions are required for different applications.

In order to enable large-scale ICT applications among vehicles, novel concepts are needed for processing, storing, and sharing data within the participating cars. This can be done most efficiently by hierarchically organizing cars into groups using vehicular cloud [19, 34] and clustering [47, 11] concepts. In short, cars close to each other form small clusters called micro clouds to enable cooperative processing of data. Such micro clouds together form much larger city-wide macro clouds, which are eventually connected to data centers in the backend [28].

In this paper, we concentrate on the data management in micro clouds. Conceptually, this is very similar to Mobile Edge Computing (MEC), even though there are completely novel challenges to be solved related to the cars' mobility. Micro clouds, i.e., clusters of cars, can be formed using both parked cars [24] and moving cars [23] alike. Here, we focus

*Corresponding author.
Email addresses: hagenauer@ccs-labs.org (Florian Hagenauer), ta-higuchi@us.toyota-itc.com (Takamasa Higuchi), onur@us.toyota-itc.com (Onur Altintas), dressler@ccs-labs.org (Falko Dressler)

Article published in Elsevier Ad Hoc Networks 91 (2019) 101871. (10.1016/j.adhoc.2019.101871)

on moving cars that form clusters at specific geographic locations [27], extending and improving our recently proposed approach [23]. In contrast to other solutions in the literature, our proposed clusters are not defined by cars being close to each other, but by cars being close to a certain location, e.g., an intersection. By putting a cluster close to the center of an intersection, we leverage direct line-of-sight communication between Cluster Members (CMs) arriving from multiple directions.

For determining the current cluster configuration, we do not rely on complex distributed algorithms, but rather exploit infrastructure such as an Access Point (AP) to perform the necessary calculations. This centralized approach provides a holistic view of all the vehicles in the vicinity. APs only need a small set of functionalities: receive/send messages from/to nodes in their surroundings. In addition, an AP may provide a direct link to the macro cloud and to additional computational resources. To gather all necessary control information, the AP periodically receives control information from all cars. Based on this, cars are distributed into clusters and informed about their roles (i.e., being Cluster Head (CH) or a CM).

Even more important is the actual data management in such micro clouds. We developed two such services to demonstrate the roles micro clouds play as part of a vehicular cloud. The first *collect* service is performing a regular data upload from the micro to the macro cloud. Its goal is to collect sensing and control data from the micro cloud, similar to Floating Car Data (FCD), e.g., geographic position, speed, direction, or temperature. Every CM sends its current data to the CH, which then forwards it to the macro cloud. The CH may also process and aggregate received data to offload resources from the macro cloud and the communication channel. The second *preserve* service aims at keeping data within a geographical context. Most of the data in vehicular networks is only relevant for a particular region and for a period of time. Therefore, there is no need to collect all data in the macro cloud. The challenge is to preserve data even though the cluster will change over time as cars join and leave. To achieve this, the data in the micro cloud is replicated to ensure it exists even after the originating car left.

Our main contributions can be summarized as follows:

1. We introduce a new concept for geo-assisted clustering based on map information. Micro clouds can thus be located directly at relevant intersections to optimize communication within the cluster.
2. We further propose and evaluate two key micro cloud services to maintain data within the micro cloud or to efficiently upload the data to the macro cloud or a backend data center, respectively.

2. Related Work

The algorithms we propose in this work are related to various concepts mostly from the domain of vehicular networking. First, we discuss work related to *vehicular clouds* to outline where our algorithm fits in the current literature. Second, we discuss the concepts we built upon for our clustering algorithms and applications; these are mostly related to clustering in vehicular networks.

2.1. Vehicular Cloud and Mobile Edge Computing

The concept of the vehicular cloud has been coined in the years 2012 and 2013 [19, 15, 41]. Generally, a vehicular cloud consists of a group of vehicles autonomously working together to coordinate and provide services and resources [7]. Clouds in this context evolved from traditional cloud computing concepts being investigated for vehicular networks. As a result, as discussed by Whaiduzzaman et al. [51], traditional cloud services (e.g., Storage as a Service, Computing as a Service) can be provided by vehicles. When comparing these vehicular clouds to the definition of traditional clouds [38], they are able to fulfill the essential characteristics (on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service). Furthermore, vehicular clouds provide opportunities for new services, e.g., Network as a Service or Cooperation as a Service [51]. A unique feature all vehicular cloud approaches have in common is their inherent mobility, which allows them to be versatile where and how to provide services but also makes these solutions more challenging due to the dynamic topologies. Gerla [19] discussed the formation of mobile clouds by cars in order to provide locally relevant services. These clouds should offload tasks from the internet cloud, which they are better suited to process. Similarly, Olariu et al. [41] exploit the mobility of vehicles to form vehicular clouds and highlight this feature as a distinction from conventional cloud computing. Nevertheless, parked cars may form a vehicular cloud and act similar to a conventional cloud [3, 14]. Such stationary clouds exploit unused resources from parked cars.

In contrast, MEC aims to move resources closer to users, i.e., to the edge of a network. It first appeared in the form of *cloudlets* aiming to place large amounts of computational resources reachable via WLAN close to end users [44]. Similarly, the concept of *fog computing* aims to extend cloud computing by providing compute, storage, and networking services at the edge of the cloud [5]. Such a platform should enable resource sharing while still maintaining location awareness and low latency. The term Mobile Edge Computing has been used, e.g., by ETSI to describe an architecture enabling mobile edge applications on top of virtual infrastructure close to or at the edge of the network [29]. Such applications benefit from resources closer to the users instead of a (potentially distant) cloud. Concrete examples from ETSI include augmented reality, improved video streaming, and connected cars [29]. Mach and Becvar [37] categorize potential MEC applications into three categories: consumer oriented services, operator and third-party services, and network performance and QoE improvement services.

To bring MEC and vehicular clouds together, a hierarchy of *micro* and *macro* clouds was introduced by Higuchi et al. [28]. While macro clouds aim to cover entire cities [2] or even interconnect them [25], micro clouds are restricted to smaller areas [19, 14]. Both types offer various services to users of the cloud, drivers and non-drivers alike. As outlined by Gerla [19], cars in a micro cloud may work together to perform computationally intensive tasks. Thus, micro clouds usually operate using local data and providing services to users in the surroundings. Depending on how such micro clouds are formed and the density of available vehicles, they might not be able to provide all classical cloud computing services outlined by NIST [38]. Therefore, they could focus on specialized services exploiting their mobility or services which are able to cope with the mobility issues. Macro clouds consist of multiple micro clouds, potentially covering whole cities. Therefore, they can be seen as a gateway to providing a much larger number of services and in turn bringing them closer to fulfilling all NIST cloud computing characteristics.

The architecture we present in this paper is based on concepts from both MEC and vehicular clouds. We outline how to form micro clouds at geographic locations and present two mobile edge services for them. These services show how such clouds provide information which is then used to support useful services to consumers and operators alike.

2.2. Clustering

Micro clouds can be formed by grouping cars based on various parameters, a process commonly referred to as clustering. Clustering is not unique to vehicular clouds, but has been generally investigated in the scope of Mobile Ad Hoc Networks (MANETs) [47] and later adapted for vehicular networks [11].

Clustering algorithms are usually quite domain specific, which particularly holds also for the vehicular use case. Thus, different solutions have been designed for use in freeway [4, 6, 31, 50] and urban [48, 49, 8, 27, 35] environments, respectively. While the traffic patterns on freeways are rather simple (few chances to enter or exit), urban environments are way more dynamic [39]. This results in rapid network fragmentation so that clustering algorithms often rely on additional infrastructure (e.g., Roadside Units (RSUs) or cellular base stations) for coordination. Furthermore, intersections and other landmarks are helpful for efficient clustering [36, 35]; cars close to these landmarks may further experience better connectivity.

Looking at clustering approaches for vehicular networks, many focus on safety applications [4, 48, 16] while others take a more generic approach [12, 20, 50, 8]. Safety applications usually aim to send warning messages to cars in close proximity (urban) [48, 16] or specifically to following cars (freeway) [4]. The more generic clustering algorithms do not rely on a certain application. Wang and Lin [50] propose a clustering process based on location, velocity, and link lifetime, which aims to provide a constant bit-rate data traffic. The algorithm is passive as the necessary information is piggy-backed on to control messages. Crepaldi et al. [12] proposed QuickSilver, an algorithm to support all kinds of applications by combining node-centric and content-centric communication. Similarly, Caballeros Morales et al. [8] developed a clustering solution based on the cars' destination, which leads to clusters traveling through the city instead of staying at a fixed location.

In summary, it can be said that the most successful clustering solutions build upon four steps [11]: First, cars (or a coordinator) gather control data from all potential CMs. Second, the current cluster configuration is generated and cars are selected as either CH or CMs. Third, the configuration is distributed to all cars. Finally, the cluster is operational and data can be gathered by the CHs from all CMs.

2.3. Data Management Services

Cars participating in vehicular networks provide capabilities comparable to mobile sensing due to their available sensors and their inherent mobility. Initially, the focus of mobile sensing was mainly on mobile phones, their applications, and the sensing scale [33]. Based on the categorization presented by Lane et al. [33], our proposed clustering approach enables *group sensing* (close-by nodes exchange sensor data) and *community sensing* (large number of participating nodes) applications.

A concrete example was proposed by Zhou et al. [52], who use mobile phones to estimate traffic density. Their approach does not require user interaction, but rather lets the phones detect if they are on a public bus. This information is then combined with cellular signal strength to estimate the position and measure the travel time. Sometimes, multiple nodes in close proximity need to perform the same sensor task, although only one node is required. Hemminki et al. [26] present a system where such close-by nodes are clustered and only one CM needs to perform the sensing. This is done based on control information shared with a central backend. Overall, the goal of these approaches is to exploit phone sensors while keeping energy consumption low. This is not a key concern for vehicular networks, which enables additional opportunities.

Based on our presented clustering concept, we propose two data management services inspired by mobile sensing, namely to *collect* data for further forwarding to the macro cloud or a data center (in potentially aggregated form) and to *preserve* data for providing a geo-coordinate-aware distributed storage. Collecting data for safety purposes has been investigated already by Raya et al. [43]. Their main goal was not to reduce the load on the channel, but rather to support secure aggregation. Another example is the traffic information system by Ibrahim and Weigle [30], which employs aggregation in dense traffic conditions. Similarly, Taherkhani and Pierre [48] collect information from cars before distributing safety messages. This is one of the more complex architectures clearly differentiating between congestion detection, cluster management, and

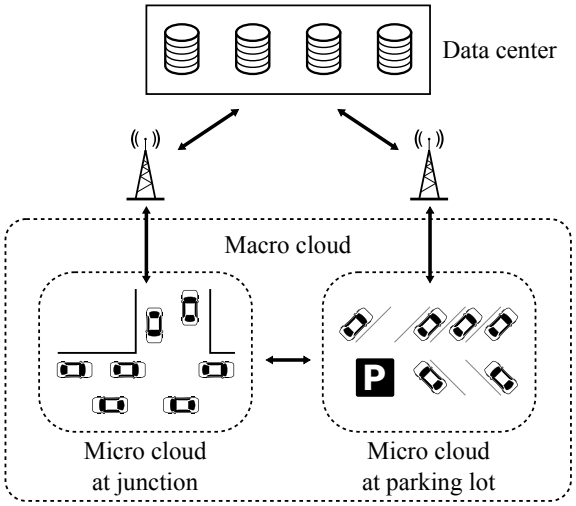

Figure 1: Overview of the vehicular cloud architecture bringing together local micro clouds, large-scale macro clouds, and back-end data centers. The uplink from micro to macro cloud and to the data center can be realized using different communication technologies.

congestion control. Landmarks have been first introduced for solving routing problems. For example, Lee et al. [35] establish connections between such landmarks to improve connectivity. Lochert et al. [36] also use landmarks to optimize RSU placement in a city. In our approach, we make use of such RSUs to form micro clouds.

The proposed *preserve* service is closely related to Floating Content (FC). Again proposed for mobile phones, Ott et al. [42] aim to keep data locally available. Based on range and interest, nodes exchange FC without any central coordination and try to keep the data alive. Still, there is no way to recover lost data, which we deem essential for urban vehicular networks and their changing densities. Ciocan et al. [10] evaluated this FC concept using connected cars in urban environments. According to their results, vehicles are able to preserve data, if the density is high enough — independent of the size of the area of interest. Nevertheless, they do not propose a recovery mechanism for low density scenarios. Generally, FC is not limited to urban scenarios, but can also be used on freeways, e.g., to share alert messages. Based on such a scenario, Nakano and Miyakita [40] investigated the effects of vehicle density and opposite traffic on the success. Based on numerical analysis, the authors conclude that these factors are very important and it is not sufficient to assume constant traffic volume and speed.

3. Architecture

Our proposed architecture aims to form micro clouds, which provide services in conjunction with large-scale macro clouds and back-end data centers. As shown in Figure 1, multiple micro clouds form a macro cloud and connect to a data center where further data processing capabilities are available. In this paper, we concentrate on two concrete services, one supporting the macro cloud in collecting location-based data, and the other preserving data locally in a micro cloud, e.g., aggregated traffic or parking information.

Both services build upon a novel form of *geographic clustering*, which we, in an early form, already outlined in [23]. Our clusters include moving cars and form around relevant locations in a city, especially around intersections. The cluster coordination is done by an AP, making the algorithm more robust using this more centralized approach. We choose this approach as APs are common in cities, especially if we consider 5G and its micro cells. Furthermore, APs are able to acquire a holistic view of all cars in the surroundings due to their coverage.

3.1. Requirements

There are a number of building blocks our architecture relies upon, all provided by participating cars and the AP:

Inter-Vehicle Communication (IVC): Every car needs to be able to communicate with other cars in the vicinity. This can be using an arbitrary Device-to-Device (D2D) communication technology such as DSRC, WLAN, or LTE-D2D, etc. In the scope of this paper, we make use of the IEEE 802.11p protocol.

Communication to Access Points: It should be possible for every car to communicate with APs in the vicinity. This can be done either by using the same technology as for IVC or by relying on a dedicated communication channel, e.g., an LTE uplink connection. In this paper, we rely, similar to IVC, on IEEE 802.11p for the communication with the AP.

Geographic Positioning: The coordination of the cluster relies on geographic position of all participating cars. Thus, these cars need to be able to determine their geographic location. For this, cars can make use of a Global Navigation Satellite System (GNSS) such as GPS, GLONASS, or Galileo.

Connection to Macro Cloud: All micro clouds need to have a communication channel to the macro cloud. While this could be realized with a cellular connection from any participating car, in our case, we assume a connection from the AP. This connection can then be used to upload the collected data (*collect* service) or request stored data (*preserve* service).

Select Cluster Locations: Our clusters are to be formed around relevant locations in a city. CH selection and successively cluster creation is based on these locations. We recommend to center them at intersections to provide the best possible connection quality between the CH and its CMs. This has been further discussed by Higuchi et al. [27].

3.2. Geographic Clustering

As outlined by Cooper et al. [11], every clustering algorithm consists of a handful of generic core procedures

Algorithm 1 Selection of CHs

Input: $\mathbb{J}$, set of all junctions.
Input: $\mathbb{C}$, set of all cars within communication range of the AP.
Input: $\mathbb{D}$, set $\{ d_{jc} \mid$ distance from junction j to car $c \}$.

```
for j ∈ J do
    Create a list of cars ordered by distance to the junction j.
    First car in the list is a candidate CH of the cluster at j.
end for
while Two junctions have the same candidate CH c do
    Remove c from the junction with the larger distance.
end while
CH = empty map of all CHs as keys, and a list of cluster members as value.
for j ∈ J do
    h = CH candidate for j.
    Add a new cluster to CH with h as a key and cluster head.
    Remove h from C.
end for
for c ∈ C do
    -> Add c as a CM of the cluster at the junction j whose distance d_jc is the smallest.
end for
```

regarding cluster formation and cluster maintenance. The algorithm we propose follows these procedures and adapts them to enable geographic clustering. Our map-based cluster solution consists of four steps, which we outline in the following.

1. *Gather Control Data*: At a fixed interval, every car broadcasts its current control information – which is in turn received by all APs in communication range. The control information includes, among others, the car's current position. Thus, standard CAMs can be exploited for this purpose.
2. *Create clusters*: Based on the received information, the APs calculate the best cluster configuration for the next time interval. Our clustering algorithm is outlined in Algorithm 1. Besides creating the clusters, cars also get assigned their cluster roles (CH or CM). Such a hierarchical structure can furthermore be exploited by services running on top of the cluster. At the beginning, a CH is selected for each cluster: First, for every junction, APs collaboratively create a list of potential CHs for each junction. The list is initialized by all the cars within the communication range of any of the APs, ordered by distance from the junction. The car closest to a junction (i.e., the first one in the list) is selected as a candidate CH for the corresponding cluster. If a car is selected candidate CH of two clusters, only the cluster closer to the car can keep it as a candidate CH. The other cluster removes the car from its list of potential CHs and selects the next closest car in the list as an alternative candidate. This is repeated until every cluster has a unique CH. Note that, if the number of cars is smaller than the number of junctions, there will be less clusters than junctions. After all CHs have been selected, all remaining cars are assigned to the clusters: every car is placed as a CM into the cluster at its closest junction.
   A car can be part of multiple clusters created by different APs. This is not an issue, rather an opportunity: it can act as a gateway between two clusters and provide cluster-spanning services. Similarly, a single AP is able to coordinate multiple clusters. This is useful if the covered area contains several potential cluster locations, e.g., an LTE eNodeB covering multiple blocks.
3. *Distribute Control Information*: After the cluster configurations have been calculated, the results are distributed by means of a broadcast from the AP to all cars. Upon receiving this information, the cars update their cluster configuration according to their role assignment.
4. *Gather Data*: This final step is to collect data from clusters [11]. Our algorithm does not specify this part, which is left to the specific services using the cluster configuration.

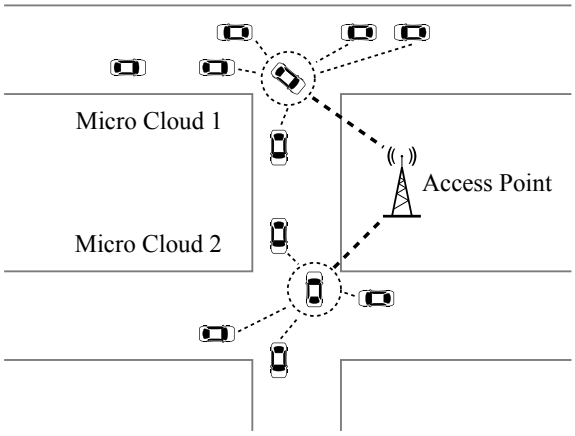

Figure 2: An example of two micro clouds formed using geographic clustering. In principle, the car closes to the junction is CH. After the CH is selected, other cars become CMs of their closest cluster.

Although we select a CH, it is not essential for the geographic clustering algorithm. We could just put all cars as CMs into the cluster they are closest to without needing any CHs. But, many services rely on a dedicated CH for their operation. Examples include the *collect service* discussed in Section 3.3 or the algorithm proposed by Tung et al. [49]. In both examples, the CHs are in charge of communicating cluster information to the AP. To support such services without any investment from their side, our

geographic clustering algorithm selects a CH when calculating the clusters. Furthermore, selecting a CH close to the intersection improves in-cluster communication between CMs and their CH for two reasons: (1) the distance between them tends to be shorter and (2) cars approaching the intersection will be in line-of-sight of the CH.

As long as a car is able to communicate with an AP, it is part of at least one cluster. The maximum size of such a cluster is limited by the transmission range of the APs, as vehicles outside of this range do not receive any cluster coordination information. If there is a gap between two APs, cars in this region will not be part of any cluster. Every car furthermore keeps track of how long it is part of a cluster, and, if there was no subsequent information, it drops out of the cluster.

An example of two clusters generated by our algorithm can be seen in Figure 2. Clusters are located at intersections with the CH being the car closest to the center. Other cars are then assigned to their closest cluster. This ensures that CMs on the approaching streets have a high chance of a line-of-sight connection to the CH.

3.3. Micro Cloud Services

The proposed clustering algorithm is the foundation for many potential services and applications running in such a micro cloud. In the following, we outline two fundamental services that support such applications, namely *collect* (collecting and aggregating data at the CH) and *preserve* (maintaining data within the local geo-context).

3.3.1. Collect Data Service

The first service collects sensor information from the CMs and sends it via the AP to the macro cloud. Such sensor information can for example be FCD and can be relevant for cars as well as for other users in the scenario. While doing this, the micro cloud is already able to process and aggregate data, which reduces processing in the macro cloud and also substantially reduces the required communication resources. We build upon our earlier work on the collection of FCD by micro clouds [23].

The *collect* service works as follows: Cars periodically collect information from various on-board sensors (e.g., positional information, weather sensors, camera pictures or even videos). This data is initially stored in a local Knowledge Base (KB).

The collected data is then sent to the CH, which adds it to its own KB. Periodically, the CH sends the content of its current KB to the AP. Note that, the CH does not need to keep any state information about the cluster. Therefore, even if the CH changes frequently due to high vehicle speeds, no handover between subsequent CHs is necessary and no significant overhead incurs. If possible, the collected data is preprocessed and aggregated to save bandwidth and reduce the load on the wireless channel. Depending on the nature of the stored data, such aggregation has the potential to reduce the amount of transferred data significantly [13]. If the aggregation takes place in the micro cloud, also computation is offloaded from the macro cloud.

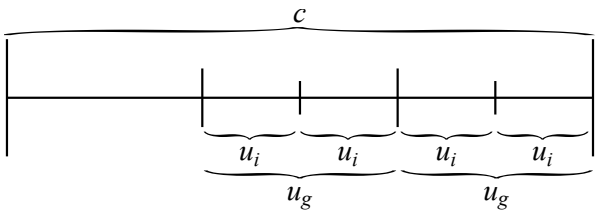

Figure 3: An example of the upload interval calculation with a CH calculation interval c, four uploads u_i, and two CHs.

In previous work [22], we found that the transmission from a CH to the AP is the most critical part. This is because, if aggregation has been applied, the complete data from a single cluster can get lost if the transmission between a CH and the AP fails. Thus, adequate countermeasures similar to Automatic Repeat Requests (ARQs) need to be used.

Furthermore, we want to support a single AP handling multiple clusters, e.g., covering multiple intersections in the surroundings. Generally, this is covered by the proposed clustering algorithm as it selects the appropriate roles but we further introduce a slotting system for uploads coordinated by the AP. The goal is to avoid packet collisions with other CHs when uploading data. Note that, this is a basic approach to avoid the hidden terminal problem between multiple CHs and in the future can be replaced, e.g., by using a RTS/CTS mechanism coordinated by the AP.

Simply put, during a CH calculation interval, a CH should wait for some time and collect data from its CMs. Afterwards, the periodic upload process starts where each CHs gets its own upload slot. This enables an AP to coordinate all clusters using one interval, instead of introducing the necessity to handle multiple such processes in parallel.

More formally, we split the CH calculation interval c into multiple general upload slots u_g each having a length of $u_g = c/(|u_g| + 1)$. During the first general slot, the CHs only collect data, while during later ones they also upload collected data to the AP. We set the number of general upload slots $|u_g| = 2$, to reduce the size of a single upload and enable potential retransmission if the first upload failed. To reduce collisions, we divide every general upload slot into individual upload slots u_i for each cluster. Such an individual slot has a length of $u_i = u_g/|\mathrm{CH}|$, based on the number of CHs coordinated by the AP. Every CHs gets assigned a general upload slot and starts its upload at the beginning of it.

Whenever a car receives cluster information, it first checks if it is a CH. Afterwards, it waits for the duration of one general upload slot and starts uploading during the second one at the beginning of its individual upload slot.

An example can be seen in Figure 3 with a CH calculation interval $c = 10\,\mathrm{s}$, two uploads during that time and two CHs. This results in the length of a general upload slot

of $|u_g| \approx 3.3\,\text{s}$ and the length of an individual upload slot of $u_i \approx 1.66\,\text{s}$. Therefore, the first CH starts the upload after 3.3 s (first individual slot), while the second one starts at 5 s (second individual slot). Each of them uploads data a second time during the third general upload slot.

3.3.2. Preserve Data Service

The second service preserves data fragments at a certain geo-location, similar to FC applications, i.e., within a micro cloud. Some application examples benefiting from such stored data fragments are local advertising [42], tourist information sharing [42], information about free parking spots [1], urban security [9], or emergency situations [9]. Note that, all these applications require only local data with no inherent need to upload it to the internet or to a macro cloud.

The general idea of the *preserve* service is to distribute fragments among all CMs so that the data remains in the cluster with a high probability even though cars are coming and going, i.e., joining and leaving the micro cloud. Thus, the micro cloud performs caching operations on behalf of all CMs as well as the macro cloud, thus, the cluster can be essentially a storage extension for the macro cloud. To preserve data within the cluster, it has to be replicated to other CMs. For any data fragment, we define a 100 % *replication rate*, if all cars within the cluster carry a replica.

We assume that all data fragments have a unique identification. This can, for example, be done by using a hash of the data contents (e.g., using a *MD5* checksum). Furthermore, all data fragments have a time to live attached, after which this data can be purged from the KB. All cars amend the periodic updates to the AP with a collection of all stored data fragment IDs. Now, the AP keeps track if data fragments are available in a cluster (*overall known data*). Furthermore, the AP monitors the fragments each car has stored (*actually known data*). Periodically, the AP calculates the missing data fragments for every car. This information is then sent via broadcast to all cars. When a car receives information about missing fragments, it requests these from the other cars using unicast. To reduce the load on the channel, cars wait for a random interval before performing this request. This is done, so that not all cars request missing fragments at the same time. These requests are fulfilled by sending the requested data via unicast. When a car leaves a cluster, it updates its local KB by removing all data fragments received from the left cluster. Over time, data fragments are created by CMs of the micro cloud and other fragments' time to live expires.

The performance of the solution can be assessed by looking at the ratio between the actually known data and the overall known data as well as by measuring the channel load to achieve the goal of a perfect data distribution. To further improve the performance of the *preserve* service, we developed three additional improvements, namely passive overhearing, continued maintenance of data in neighboring clusters as well as balanced requests for missing data.

Overhear: In the basic version of the algorithm, missing fragments are sent via unicast. As a consequence, if multiple CMs are requesting the same data fragment, multiple transmissions are needed. This adds load on the wireless channel, which, in turn, could overload the channel leading to more missing fragments. As the name indicates, *overhear* allows overhearing the response messages to realize a multicast transmission. In addition, CMs aggregate multiple requests for data fragments within a time interval so that only a single reply becomes necessary.

Conserve: In the initial version of the algorithm, the local KB is cleaned up when a car leaves the cluster. This cleanup in combination with a small number of CMs may lead to more and more data fragments getting lost over time. This is especially relevant for very low car densities. *Conserve* allows cars to not clean up valid data fragments after leaving a cluster. If the car becomes a CM of a nearby cluster, these 'old' fragments are now also marked as missing for other CMs and in turn can be requested by other cluster members. This way, these fragments can be either brought back to the original micro cloud by a car or simply be overheard by members of that cluster.

Balanced Requests: A final improvement covers how cars send requests for missing data fragments. Originally, every car sent a single request for every data fragment it did not yet store in its KB. This leads to an unnecessary overhead due to messages sent from CMs. A naïve approach would be to combine requests after reception. However, this may lead to significant loss if the transmission is not successful and the destination stores a significant number of data fragments in the cluster – the destination may move away and in the meantime no requests to other destinations are placed. To mitigate this, we changed the approach to send the requests in a more balanced way. Therefore, cars aim to request the same amount of fragments from every other CM. This change makes the process of requesting data less error prone compared to the previous approach while still having less overhead compared to requesting every single fragment.

4. Evaluation

4.1. Simulation Setup

As a simulation tool, we use *Veins LTE* [21], which is based on the widespread Veins simulation framework [46]. Veins couples to the road traffic simulator SUMO[1] with the network simulator OMNeT++[2]. It also provides all necessary modules for evaluating IEEE 802.11p based DSRC communication in vehicular environments. Veins LTE further adds LTE communication to the mix, which eases the addition of later extensions, e.g., adding cellular connections to the data center.

[1] `http://sumo.dlr.de`
[2] `http://www.omnetpp.org`

For our evaluation of the geographic clustering algorithm and the two services we introduced, we used a Manhattan grid scenario as depicted in Figure 4. All metrics were collected only for the middle junction j_0 in order to avoid border effects. All the other four intersections j_{1-4} were also used for clustering. Road traffic was simulated in a much larger area around the evaluated junctions, again to avoid border effects. We simulated all parts of the proposed algorithm including control and data messages needed by the clustering algorithm and the services. Note that, while we build upon IEEE 802.11p, we do not use the ETSI ITS-G5 stack. This includes the control beacons, which are not CAM messages but, in theory, could be built upon them.

Furthermore, we used three different traffic densities to investigate the behavior of our algorithms. For low vehicle density, we often observe zero cars at the cluster location (on average two cars), thus, leading to situations in which the micro cloud cannot be established. Higher car densities usually have a sufficiently large number of cars in the environment to continuously establish a cluster; for medium car density we have about four cars on average and for a high car density we see on average nine cars with spikes of up to 80 cars.

The most general simulation parameters are summarized in Table 1. All results presented in the following are based on multiple repetitions for high statistical evidence, the detailed numbers can be found in the respective tables. Please note that, error bars represent the mean results ± standard deviation.

4.2. Cluster Structure

We first investigate the structure of the clusters. In Figure 5, we plot a histogram depicting how often cars take the role of a Cluster Member or a Cluster Head for each of the three traffic densities (all simulation specific parameters are summarized in Table 2). As the first observation, we can see cars being longer in their role as a CM compared to being a CH. This is because only the car closest to the intersection becomes CH. For this reason, we also see the difference in the duration of being a CH for the different densities: For medium and high traffic densities, the chance for finding a new CH is higher. We also see the effects of increasing traffic when we look at cars being CMs. Cars are longer part of a cluster in the higher density scenarios because the slower traffic leads to jams before intersections.

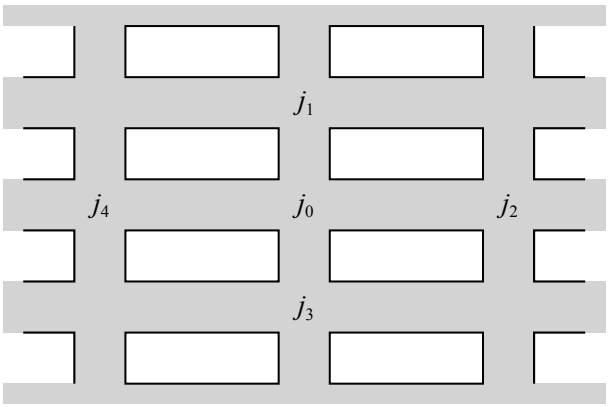

Figure 4: Manhattan grid scenario used for all evaluations. All metrics were collected for the cluster on junction j_0. The other intersections j_{1-4} participated in the clustering process, but were not considered for the evaluation.

Parameter	Value
IVC technology	IEEE 802.11p
Channel	5.89 GHz
Transmission power	20 mW
Bandwidth	10 MHz
Investigated area	800 m × 250 m
Vehicles per cluster (low)	2 (mean), 8 (max)
Vehicles per cluster (medium)	4 (mean), 17 (max)
Vehicles per cluster (high)	9 (mean), 80 (max)
Maximum vehicle speed	50 km/h

Table 1: General Simulation Parameters.

4.3. Collect Service

Looking at the performance of the *collect* service, we build upon results reported in some of our recent work [23]. In that paper, we discovered that the service is able to perform well, but degrades if data gets lost on the last hop from CH to the AP. In this paper, we are able to confirm this based on a more realistic setup with multiple traffic densities. Furthermore, we provide additional insights into the performance of the *collect* service. All service specific parameters are listed in Table 3.

Our main metric is the success rate of the service. Given d_g, the amount of data generated by all cars in a cluster and d_r, the data received by the AP, we define the success rate as d_r/d_g. Beside the traffic density, there are two main parameters that help better understand the performance of the service:

- Aggregation factor: The aggregation factor determines how much data is aggregated by the CH before sending it to the AP. An aggregation factor of 0.6 indicates that the sent data is 60 % of the size of all entries inside the KB.
- Retransmissions: In addition to the basic configuration, where data messages from CH to AP are

Parameter	Value
Data size	1 kByte on joining
Data TTL	300 s
Information collection interval	1 s
Cluster calculation interval	5 s
Repetitions	100 per configuration

Table 2: Cluster structure simulation parameters.

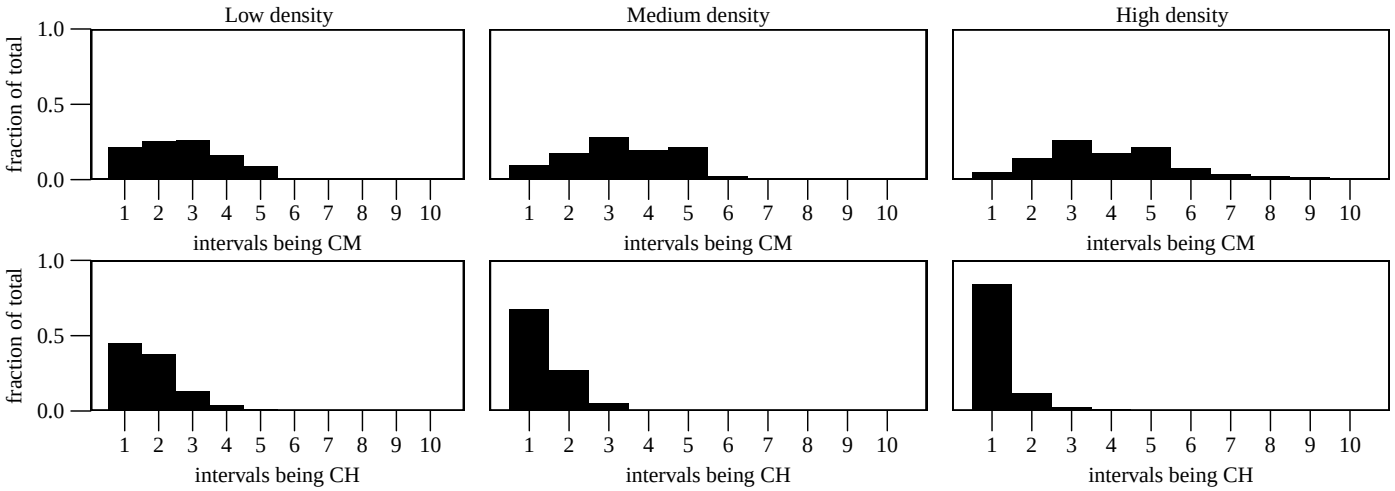

Figure 5: Histogram depicting how often cars take the role of a Cluster Member or a Cluster Head on average for the three different densities. The 0.95 confidence intervals for the shown data are ±1 %.

sent via broadcast, we also investigate the case if these messages are sent using unicast transmissions. For this, we are relying on the retransmission mechanism from IEEE 802.11p, which, beside having certain limitations in vehicular environments [32], aims to retransmit a unicast packet up to 7 times.

The results are plotted in Figure 6, where we show the mean success rate for the discussed parameters together with the standard deviation. As can be seen, the *collect* service works extremely well for a low traffic density. Some data loss can be observed when using broadcast for data transmissions, but generally the success rate is well above 90 % for all aggregation factors. When considering medium traffic density, the success rate is slightly lower, but still in the range of 75–100 %. Retransmissions again improve the performance in medium density. Finally, for a high traffic density, we can report two core observations: (1) retransmissions improve the performance significantly, and (2) aggregation is an important aspect. We can see that, even in the worst case, the scenarios with retransmissions result in higher performance. Even in case of a high traffic density and an aggregation factor 1, retransmissions are still able to increase the success from roughly 40 % to 60 %.

One of the reasons for the performance degradation for high traffic density scenarios can be seen in Figure 7. Here, we can see the mean number of received errors with the standard deviation for the traffic densities with aggregation factor 1. An error occurs if a wireless frame cannot be decoded, e.g., due to an ongoing transmission or a bit error. While there is a slight increase in errors from low to medium traffic density, a much higher increase can be observed from medium to high density. The transmissions from CHs to the AP are more likely to result in failure because of the larger packet size. With retransmissions, the number of overall transmissions increases, which leads to more errors. However, as we can see by returning to Figure 6, the performance is still better due to the successful retransmissions.

4.4. Preserve Service

To evaluate the *preserve* service, we focus on two metrics: the fraction of known data fragments and channel load. Before the AP calculates a new cluster configuration, it records the number of data fragments that should be known by at least one car in the cluster (i.e., the overall known data fragments). This number consists of fragments fulfilling two conditions: first, a CM has informed the server that it has the fragment stored in its KB and, second, the Time to Live (TTL) of the data fragment has not yet expired. In addition, the AP records the number of fragments actually known in the cluster, i.e., the currently known fragments. The currently known data fragments divided by the overall known fragments gives the fraction of known data fragments describing how successful data was preserved inside the cluster. The second metric is the load

Parameter	Value
CM to CH data	10 kByte every 2 s
Aggregation factors	0.2, 0.4, 0.6, 0.8, 1
Information collection interval	1 s
Cluster calculation interval	5 s
Repetitions	15

Table 3: Collect service simulation parameters.

Parameter	Value
Data size	1 kByte on joining
Data TTL	300 s
Information collection interval	1 s
CH interval	5 s
Repetitions	10 per scenario

Table 4: Preserve service simulation parameters.

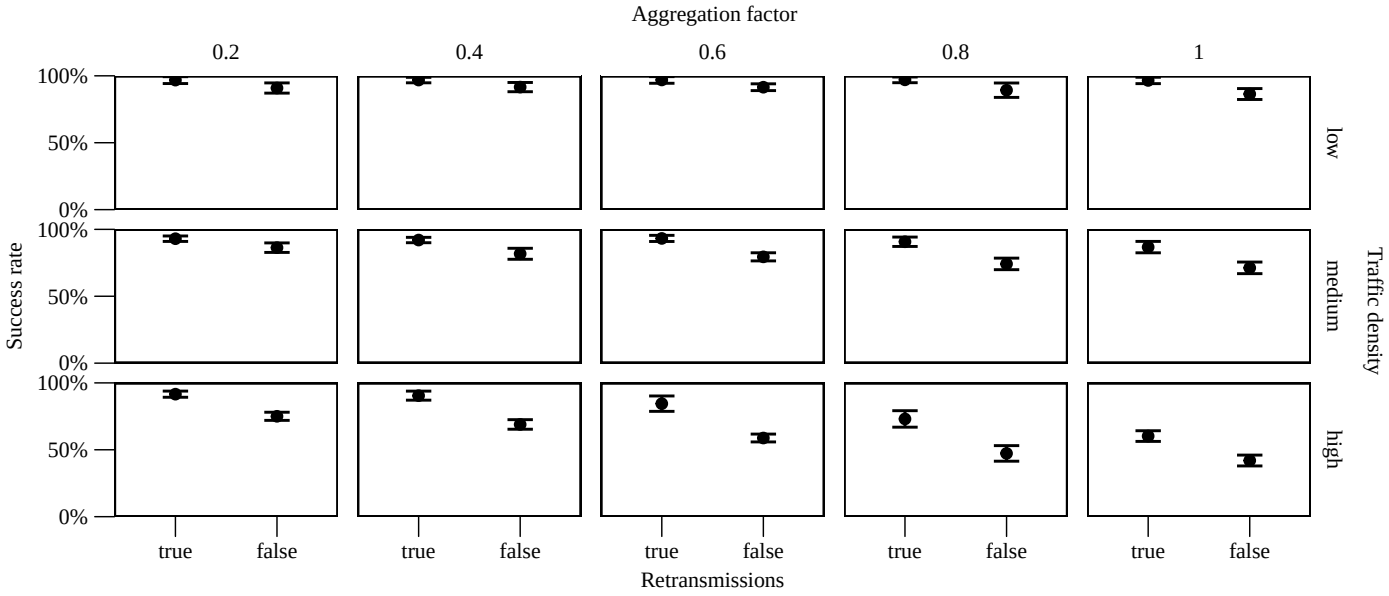

Figure 6: Mean success rate (and standard deviation) of the *collect* service with and without retransmissions for different aggregation factors and traffic densities. The 0.95 confidence interval of the shown data is on average ±2 % with a maximum of ±3 %.

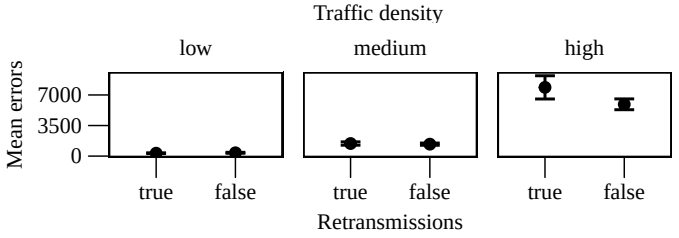

Figure 7: Mean received erroneous frame count (and standard deviation) of the *collect* service for an aggregation factor of 1 and all three traffic densities. The means and 0.95 confidence intervals of the shown data are as follows: 330 (±20)/380 (±25) errors for low density, 1440 (±90)/1370 (±60) errors for medium density, and 7870 (±660)/5930 (±300) errors for high density.

on the channel based on the MAC busy fraction observed by the AP. If this load gets too high, messages cannot be transmitted anymore and/or larger delays occur. Relevant simulation parameters are in Table 4.

For our evaluation, we set the TTL of data fragments to 300 s. This enables a car to deploy its data fragments at multiple micro clouds it passes by and, in turn, enables the macro clouds to maintain this data for a suitable time after the original car left the scenario. Note that we used 10 route configurations for each scenario. This allows to better avoid simulation artifacts.

The results regarding the fraction of known data are plotted in Figure 8. To better explain the results, we also investigate the mean channel load in Figure 9. The first row shows the baseline performance without any of our improvements. We can observe a relatively low amount of known data. For low and medium densities, this is because the cluster becomes empty or has a small number of CMs. If this happens, the stored data gets lost and can rarely be acquired again. In case of high density, cars do not receive missing data due the high channel load.

The second row shows results of the *conserve* improvement where data from other clusters is not discarded, but is actively shared. For low and medium density, this brings a clear improvement, while it fails to provide an improvement for the high density. This results from the high channel load causing interference and collisions.

The third row shows the effect of the *overhear* improvement. The highest improvements can be observed for medium and high traffic densities. While it does not perform as good as the *conserve* improvement, the load on the channel is lower. Especially for high traffic density, we see a performance boost.

The fourth row shows the *balance* improvement where requests are sent in a balanced manner to as many CMs as possible. Compared to the baseline, it is more effective, especially for the high density scenario. As we show later in this section, this approach is more effective when combined with other approaches.

We finally show the complete *preserve* service including all improvements in the fifth row. In scenarios with low traffic density, the mean known data fraction is close to 100 %. It happens rarely that a cluster looses all replicas of data fragments due to an insufficient number of cluster members. Similarly, with medium and high traffic density, basically no data fragments get lost. This comes, however, at a cost: a higher load on the channel, especially with high density, leaves less channel resources for other applications. The main factor here is the conserve improvement. To reduce this, we show in the last row both the *overhear* and the *balance* improvement enabled and the *conserve* improvement disabled. While this improves the performance in the low and medium density scenarios, it does

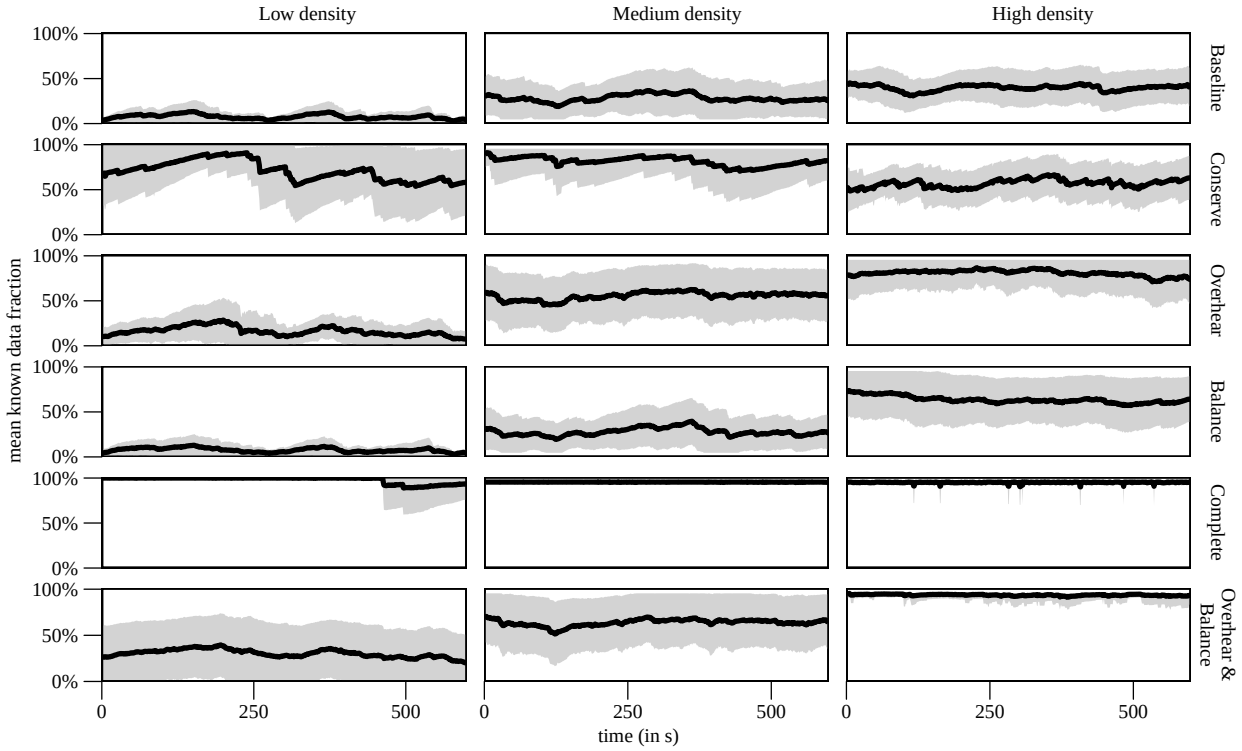

Figure 8: Success of the *preserve* service for different traffic densities: mean (black) and the standard deviation (gray). The average 0.95 confidence intervals for the shown data are (1) for low density ±2–6 %, except for the conserve improvement, where it is ±12 % and (2) for medium density ±1–6 %, except for the conserve improvement, where it is ±11 %, and, finally, (3) for high density ±1–6 %, except for the conserve improvement, where it is ±9 %.

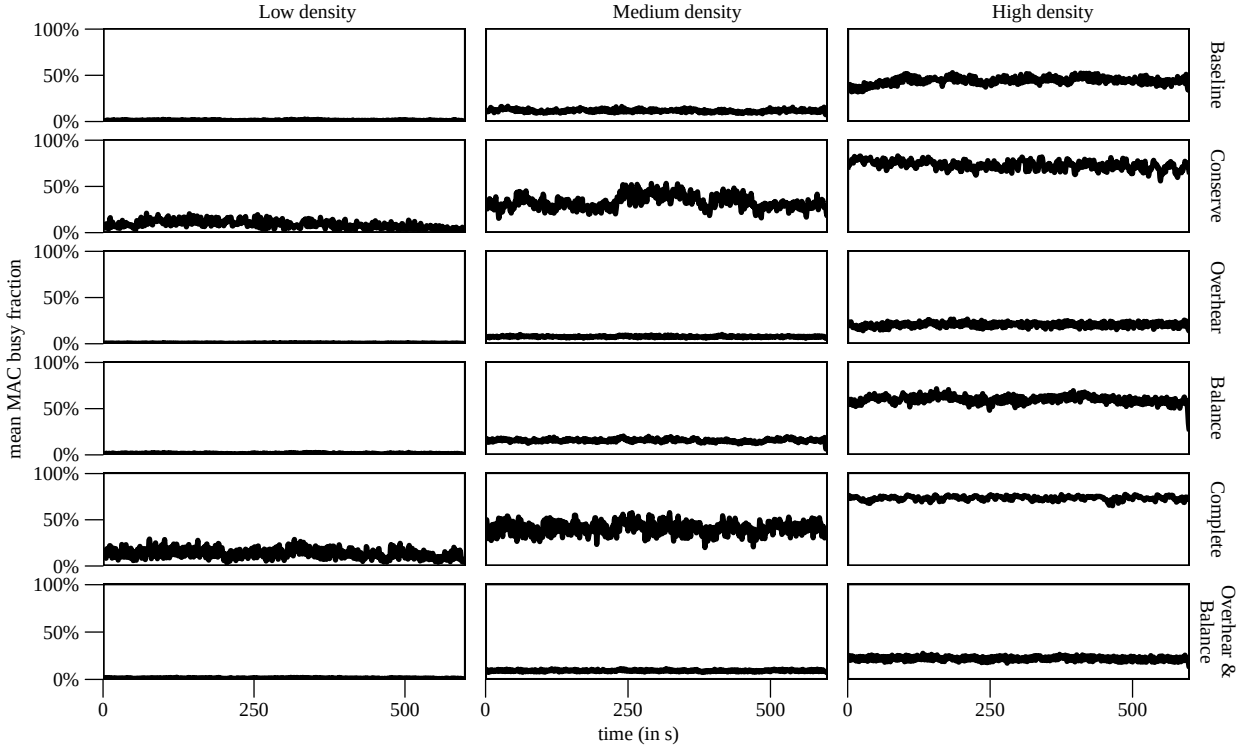

Figure 9: Mean channel load of the *preserve* service and the effect of the improvements for the traffic densities. The average 0.95 confidence intervals for the shown data are (1) for low density ±1–5 %, except for the complete improvements, where it is ±8 % and (2) for medium density ±2–4 %, except for the conserve improvement, where it is ±13 % and the complete improvements, where it is ±14 %, and finally (3) for high density ±4–8 %.

not achieve the same performance compared to the combination of all improvements. However, for the high density scenario this leads to a low channel load (roughly 20 %) while still maintaining a high known data fraction. The results show the need for APs to adapt their configuration according to the channel load. If the channel load is too high, the *conserve* improvement needs to be disabled. We envision that the similar mechanism can be also applied to vehicular micro clouds by parked cars [24].

5. Conclusion

In this paper, we presented an algorithmic solution to construct vehicular micro clouds. Our algorithm exploits existing wireless infrastructure (e.g., Wi-Fi APs or cellular base stations) to coordinate these clouds at predefined geographic locations. These APs need only a small set of features to calculate the clusters, while other features, e.g., an uplink to the internet, can be useful for further services, but are not necessary. Beside the basic clustering algorithm, we developed two fundamental services for such vehicular micro clouds. The first *collect* service allows efficient collection of sensor data from vehicles to a data center to enable large scale analysis. The second *preserve* service maintains data in the micro cloud without uploading it to the macro cloud, which helps offloading load from the macro cloud and the communication system. Furthermore, we discussed multiple improvements to further enhance the service performance. We evaluated the underlying algorithms in realistic traffic scenarios and are able to clearly demonstrate the effectiveness of our approach including all the enhancements described in the paper. Based on our approach, there are multiple interesting future research directions. As there exist numerous approaches for privacy-preserving communication in vehicular networks, it could be useful to investigate our proposed services with a focus on privacy and security. Furthermore, the concept can be extended to freeways. Hereby, vehicles move much faster, which leads to a lot of new challenges for clustering and the services.

References

[1] Shahzad Ali, Gianluca Rizzo, Balaji Rengarajan, and Marco Ajmone Marsan. 2013. A Simple Approximate Analysis of Floating Content for Context-aware Applications. In *Fourteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '13)*. ACM, Bangalore, India, 271–276. https://doi.org/10.1145/2491288.2491321

[2] Onur Altintas, Falko Dressler, Florian Hagenauer, Makiko Matsumoto, Miguel Sepulcre, and Christoph Sommer. 2015. Making Cars a Main ICT Resource in Smart Cities. In *34th IEEE Conference on Computer Communications (INFOCOM 2015), International Workshop on Smart Cities and Urban Informatics (SmartCity 2015)*. IEEE, Hong Kong, China, 654–659. https://doi.org/10.1109/INFCOMW.2015.7179448

[3] Samiur Arif, Stephan Olariu, Jin Wang, Gongjun Yan, Weiming Yang, and Ismail Khalil. 2012. Datacenter at the Airport: Reasoning about Time-Dependent Parking Lot Occupancy. *IEEE Transactions on Parallel and Distributed Systems* 23, 11 (Nov. 2012), 2067–2080. https://doi.org/10.1109/TPDS.2012.47

[4] Rachad Atat, Elias Yaacoub, Mohamed-Slim Alouini, and Fethi Filali. 2012. Delay Efficient Cooperation in Public Safety Vehicular Networks using LTE and IEEE 802.11p. In *9th Annual IEEE Consumer Communications and Networking Conference (CCNC 2012)*. IEEE, Las Vegas, NV, 316–320. https://doi.org/10.1109/CCNC.2012.6181109

[5] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. 2012. Fog Computing and Its Role in the Internet of Things. In *1st Workshop on Mobile Cloud Computing (MCC 2012)*. ACM, Helsinki, Finland, 13–16. https://doi.org/10.1145/2342509.2342513

[6] Luciano Bononi and Marco Di Felice. 2007. A Cross Layered MAC and Clustering Scheme for Efficient Broadcast in VANETs. In *4th IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS 2007)*. Pisa, Italy. https://doi.org/10.1109/MOBHOC.2007.4428735

[7] Azzedine Boukerche and Robson E. De Grande. 2018. Vehicular Cloud Computing: Architectures, Applications, and Mobility. *Elsevier Computer Networks* 135 (April 2018), 171–189. https://doi.org/10.1016/j.comnet.2018.01.004

[8] Mildred M. Caballeros Morales, Choong Seon Hong, and Young-Cheol Bang. 2011. An Adaptable Mobility-Aware Clustering Algorithm in vehicular networks. In *13th Asia-Pacific Network Operations and Management Symposium (APNOMS 2011)*. IEEE, Taipei, Taiwan. https://doi.org/10.1109/APNOMS.2011.6077004

[9] Alfredo A. Villalba Castro, Giovanna Di Marzo Serugendo, and Dimitri Konstantas. 2008. Hovering Information – Self-Organizing Information that Finds its Own Storage. In *International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (sutc 2008)*. IEEE, Taichung, Taiwan, 111–145. https://doi.org/10.1109/SUTC.2008.62

[10] Mihai Ciocan, Ciprian Dobre, Valentin Cristea, Constandinos X. Mavromoustakis, and George Mastorakis. 2014. Analysis of Vehicular Storage and Dissemination Services based on Floating Content. In *6th International Conference on Mobile Networks and Management (MONAMI 2014)*. Springer, Würzburg, Germany, 387–400. https://doi.org/10.1007/978-3-319-16292-8_28

[11] Craig Cooper, Daniel Franklin, Montserrat Ros, Farzad Safaei, and Mehran Abolhasan. 2017. A Comparative Survey of VANET Clustering Techniques. *IEEE Communications Surveys & Tutorials* 19, 1 (Feb. 2017), 657–681. https://doi.org/10.1109/COMST.2016.2611524

[12] Riccardo Crepaldi, Mehedi Bakht, and Robin Kravets. 2012. QuickSilver: Application-driven Inter- and Intra-cluster Communication in Vanets. In *Third ACM International Workshop on Mobile Opportunistic Networks (MobiOpp '12)*. ACM, Zurich, Switzerland, 69–76. https://doi.org/10.1145/2159576.2159591

[13] Stefan Dietzel, Jonathan Petit, Frank Kargl, and Björn Scheuermann. 2014. In-Network Aggregation for Vehicular Ad Hoc Networks. *IEEE Communications Surveys & Tutorials* 16, 4 (April 2014), 1909–1932. https://doi.org/10.1109/COMST.2014.2320091

[14] Falko Dressler, Philipp Handle, and Christoph Sommer. 2014. 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)*. ACM, Philadelphia, PA, 11–18. https://doi.org/10.1145/2633661.2633671

[15] Falko Dressler, Hannes Hartenstein, Onur Altintas, and Ozan K. Tonguz. 2014. Inter-Vehicle Communication – Quo Vadis. *IEEE Communications Magazine* 52, 6 (June 2014), 170–177. https://doi.org/10.1109/MCOM.2014.6829960

[16] Efi Dror, Chen Avin, and Zvi Lotker. 2012. Fast randomized algorithm for 2-hops clustering in vehicular ad-hoc networks. *Elsevier Ad Hoc Networks* 11, 7 (Sept. 2012), 2002–2015. https://doi.org/10.1016/j.adhoc.2012.02.006

[17] ETSI. 2014. *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service*. EN 302 637-2 V1.3.2. ETSI.

[18] European Commission. 2015. eCall in all new cars from April 2018. https://ec.europa.eu/digital-single-market/news/ecall-all-new-cars-april-2018

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

[20] R.T. Goonewardene, F.H. Ali, and E. Stipidis. 2009. Robust mobility adaptive clustering scheme with support for geographic routing for vehicular ad hoc networks. *IET Intelligent Transport Systems* 3, 2 (June 2009), 148–158. https://doi.org/10.1049/iet-its:20070052

[21] Florian Hagenauer, Falko Dressler, and Christoph Sommer. 2014. A Simulator for Heterogeneous Vehicular Networks. In *6th IEEE Vehicular Networking Conference (VNC 2014), Poster Session*. IEEE, Paderborn, Germany, 185–186. https://doi.org/10.1109/VNC.2014.7013339

[22] Florian Hagenauer, Christoph Sommer, Takamasa Higuchi, Onur Altintas, and Falko Dressler. 2017. Parked Cars as Virtual Network Infrastructure: Enabling Stable V2I Access for Long-Lasting Data Flows. 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)*. ACM, Snowbird, UT, 57–64. https://doi.org/10.1145/3131944.3131952

[23] Florian Hagenauer, Christoph Sommer, Takamasa Higuchi, Onur Altintas, and Falko Dressler. 2017. 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)*. ACM, Snowbird, UT, 31–35. https://doi.org/10.1145/3131944.3133937

[24] Florian Hagenauer, Christoph Sommer, Takamasa Higuchi, Onur Altintas, and Falko Dressler. 2018. Vehicular Micro Cloud in Action: On Gateway Selection and Gateway Handovers. *Elsevier Ad Hoc Networks* 78 (Sept. 2018), 73–83. https://doi.org/10.1016/j.adhoc.2018.05.014

[25] Florian Hagenauer, Christoph Sommer, Ryokichi Onishi, Matthias Wilhelm, Falko Dressler, and Onur Altintas. 2016. Interconnecting Smart Cities by Vehicles: How feasible is it?. In *35th IEEE Conference on Computer Communications (INFOCOM 2016), International Workshop on Smart Cities and Urban Computing (SmartCity 2016)*. IEEE, San Francisco, CA, 788–793. https://doi.org/10.1109/INFCOMW.2016.7562184

[26] Samuli Hemminki, Kai Zhao, Aaron Yi Ding, Martti Rannanjärvi, Sasu Tarkoma, and Petteri Nurmi. 2013. CoSense: A Collaborative Sensing Platform for Mobile Devices. In *11th ACM Conference on Embedded Networked Sensor Systems (SenSys '13)*. ACM, Roma, Italy. https://doi.org/10.1145/2517351.2517402

[27] Takamasa Higuchi, Falko Dressler, and Onur Altintas. 2018. How to Keep a Vehicular Micro Cloud Intact. In *87th IEEE Vehicular Technology Conference (VTC2018-Spring)*. IEEE, Porto, Portugal. https://doi.org/10.1109/VTCSpring.2018.8417759

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

[29] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher, and Valerie Young. 2015. *Mobile Edge Computing - A key technology towards 5G*. White Paper No. 11. ETSI. https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp11_mec_a_key_technology_towards_5g.pdf

[30] Khaled Ibrahim and Michele C. Weigle. 2008. CASCADE: Cluster-Based Accurate Syntactic Compression of Aggregated Data in VANETs. In *IEEE Global Telecommunications Conference (GLOBECOM 2008)*. IEEE, New Orleans, LA. https://doi.org/10.1109/GLOCOMW.2008.ECP.59

[31] Sanaz Khakpour, Richard W. Pazzi, and Khalil El-Khatib. 2017. Using clustering for target tracking in vehicular ad hoc networks. *Elsevier Vehicular Communications* 9 (July 2017), 83–96. https://doi.org/10.1016/j.vehcom.2017.02.002

[32] Florian Klingler, Falko Dressler, and Christoph Sommer. 2018. The Impact of Head of Line Blocking in Highly Dynamic WLANs. *IEEE Transactions on Vehicular Technology* 67, 8 (Aug. 2018), 7664–7676. https://doi.org/10.1109/TVT.2018.2837157

[33] Nicholas D. Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, and Andrew T. Campbell. 2010. A Survey of Mobile Phone Sensing. *IEEE Communications Magazine* 48, 9 (Sep. 2010), 140–150. https://doi.org/10.1109/MCOM.2010.5560598

[34] Euisin Lee, Eun-Kyu Lee, Mario Gerla, and S.Y. Oh. 2014. Vehicular Cloud Networking: Architecture and Design Principles. *IEEE Communications Magazine* 52, 2 (Feb. 2014), 148–155. https://doi.org/10.1109/MCOM.2014.6736756

[35] Kevin C. Lee, Michael Le, Jérôme Härri, and Mario Gerla. 2008. LOUVRE: Landmark Overlays for Urban Vehicular Routing Environments. In *68th IEEE Vehicular Technology Conference (VTC2008-Fall)*. IEEE, Calgary, Canada, 2157–2162. https://doi.org/10.1109/VETECF.2008.447

[36] Christian Lochert, Björn Scheuermann, Christian Wewetzer, Andreas Luebke, and Martin Mauve. 2008. Data Aggregation and Roadside Unit Placement for a VANET Traffic Information System. In *5th ACM International Workshop on Vehicular Inter-Networking (VANET 2008)*. ACM, San Francisco, CA, 58–65. https://doi.org/10.1145/1410043.1410054

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

[38] Peter Mell and Tim Grance. 2011. *The NIST Definition of Cloud Computing*. Standard Special Publication 800-145. National Institute of Standards and Technology.

[39] Diala Naboulsi and Marco Fiore. 2017. Characterizing the Instantaneous Connectivity of Large-Scale Urban Vehicular Networks. *IEEE Transactions on Mobile Computing* 16, 5 (May 2017), 1272–1286. https://doi.org/10.1109/TMC.2016.2591527

[40] Keisuke Nakano and Kazuyuki Miyakita. 2016. Information floating on a road with different traffic volumes between opposite lanes. *Japan Society for Simulation Technology Journal of Advanced Simulation in Science and Engineering* 3, 1 (March 2016), 97–113. https://doi.org/10.15748/jasse.3.97

[41] Stephan Olariu, Tihomir Hristov, and Gongjun Yan. 2013. The Next Paradigm Shift: From Vehicular Networks to Vehicular Clouds. In *Mobile Ad Hoc Networking*. Wiley, 645–700. https://doi.org/10.1002/9781118511305.ch19

[42] Jörg Ott, Esa Hyytiä, Pasi Lassila, Tobias Vaegs, and Jussi Kangasharju. 2011. Floating Content: Information Sharing in Urban Areas. In *IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, Seattle, WA, USA, 136–146. https://doi.org/10.1109/PERCOM.2011.5767578

[43] Maxim Raya, Adel Aziz, and Jean-Pierre Hubaux. 2006. Efficient Secure Aggregation in VANETs. In *3rd International Workshop on Vehicular Ad Hoc Networks (VANET '06)*. ACM, Los Angeles, CA, 67–75. https://doi.org/10.1145/1161064.1161076

[44] Mahadev Satyanarayanan, Paramvir Bahl, Ramon Caceres, and Nigel Davies. 2009. The Case for VM-Based Cloudlets in Mobile Computing. *IEEE Pervasive Computing* 8, 4 (Oct. 2009), 14–23. https://doi.org/10.1109/MPRV.2009.82

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

[46] Christoph Sommer, Reinhard German, and Falko Dressler. 2011. Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. *IEEE Transactions on Mobile Computing* 10, 1 (Jan. 2011), 3–15. https://doi.org/10.1109/

TMC.2010.133

[47] Victor Sucasas, Ayman Radwan, Hugo Marques, Jonathan Rodriguez, Seiamak Vahid, and Rahim Tafazolli. 2016. A survey on clustering techniques for cooperative wireless networks. *Elsevier Ad Hoc Networks* 47 (Sept. 2016), 53–81. https://doi.org/10.1016/j.adhoc.2016.04.008

[48] Nasrin Taherkhani and Ssamuel Pierre. 2016. Centralized and Localized Data Congestion Control Strategy for Vehicular Ad Hoc Networks Using a Machine Learning Clustering Algorithm. *IEEE Transactions on Intelligent Transportation Systems* 17, 11 (Nov. 2016), 3275–3285. https://doi.org/10.1109/TITS.2016.2546555

[49] Lung-Chih Tung, Jorge Mena, Mario Gerla, and Christoph Sommer. 2013. A Cluster Based Architecture for Intersection Collision Avoidance Using Heterogeneous Networks. In *12th IFIP/IEEE Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net 2013)*. IEEE, Ajaccio, Corsica, France. https://doi.org/10.1109/MedHocNet.2013.6767414

[50] Sheng-Shih Wang and Yi-Shiun Lin. 2013. PassCAR: A passive clustering aided routing protocol for vehicular ad hoc networks. *Elsevier Computer Communications* 36, 2 (Jan. 2013), 170–179. https://doi.org/10.1016/j.comcom.2012.08.013

[51] Md Whaiduzzaman, Mehdi Sookhak, Abdullah Gani, and Rajkumar Buyya. 2014. A survey on vehicular cloud computing. *Elsevier Journal of Network and Computer Applications* 40 (April 2014), 325–344. https://doi.org/10.1016/j.jnca.2013.08.004

[52] Pengfei Zhou, Yuanqing Zheng, and Mo Li. 2012. How Long to Wait?: Predicting Bus Arrival Time with Mobile Phone Based Participatory Sensing. In *10th International Conference on Mobile Systems, Applications, and Services (MobiSys '12)*. ACM, Low Wood Bay, Lake District, UK, 379–392. https://doi.org/10.1145/2307636.2307671