HEINZ NIXDORF INSTITUT
UNIVERSITÄT PADERBORN

Florian Hagenauer

# An Architecture for Connected Cars Providing Virtual Infrastructure and Services

Dissertation

March 2019

Please cite as:
Florian Hagenauer, “An Architecture for Connected Cars Providing Virtual Infrastructure and Services,”
Ph.D.Thesis (Dissertation), Heinz Nixdorf Institute, Paderborn University, Germany, June 2019.


Distributed Embedded Systems (CCS Labs)
Heinz Nixdorf Institute, Paderborn University, Germany

Fürstenallee 11 · 33102 Paderborn · Germany

http://www.ccs-labs.org/

# An Architecture for Connected Cars Providing Virtual Infrastructure and Services

Dissertation
zur Erlangung des Grades

DOKTOR DER NATURWISSENSCHAFTEN

vorgelegt von

**Florian Hagenauer**

geb. am 11. August 1989
in Salzburg, Österreich

angefertigt in der Fachgruppe

**Distributed Embedded Systems**
**(CCS Labs)**

**Heinz Nixdorf Institut**
**Universität Paderborn**

| | |
|---|---|
| Betreuer: | **Prof. Dr.-Ing. habil. Falko Dressler** |
| Gutachter: | **Prof. Dr.-Ing. habil. Falko Dressler** |
| | **Marco Fiore, Ph.D.** |
| Tag der Abgabe: | **05. März 2019** |
| Tag der Promotion: | **13. Juni 2019** |

# Abstract

Since multiple decades, smart cities are envisioned to provide benefits to their inhabitants, e.g., an improved quality of life, economic growth, and even a sustainable environment. These benefits are enabled by a wide range of building blocks, including Information and Communication Technologies as well as smart mobility concepts. Both of them can be provided by connected cars. Such vehicles with communication, processing, and storage resources are currently introduced into the market. In this Ph.D. thesis, we propose two ways in which connected cars are able to show their potential. First, by using them to provide efficiency and infotainment services for drivers and non-drivers alike. Second, by making these cars replace and complement communication infrastructure, i.e., by providing *Virtual Infrastructure*. A core idea is to group connected cars in proximity and form so-called Vehicular Clouds. Our contributions answer research questions regarding cloud access, service discovery and use, as well as resource utilization. In our first contribution, we investigate connected cars as an information hub for smart cities. We propose an architecture enabling the discovery and usage of services. The evaluation indicates that our presented approaches work well, even with a low penetration rate of equipped vehicles. Furthermore, we extend the concept to freeways by adding support for long-distance service discovery. The second contribution discusses how cars can be used to provide virtual networking infrastructure. We explore how vehicular clouds, formed along streets and in parking lots, enable long-lasting connections to passing cars. Finally, in our third contributions, we investigate clouds formed using moving cars and how they manage data. Beside algorithms used for forming these clouds at geographic locations, we outline two core data management services to offload storage from a data center and collect aggregated data. Throughout our investigation, we discuss multiple improvements for these services, reducing their resource usage and enhancing their performance. To summarize, the presented architectures show that resources from connected cars are well-suited to support users in discovering and utilizing services. Further, parked connected cars are, due to the resources they can provide, indeed an alternative to building new infrastructure.

# Kurzfassung

Seit mehreren Jahrzehnten gibt es die Idee von Smart Cities die das Leben ihrer Bewohner auf verschiedene Weisen verbessern sollen. Sie sollen unter anderem für eine verbesserte Lebensqualität sorgen, Wirtschaftswachstum fördern und eine nachhaltigere Umwelt ermöglichen. Diese Verbesserungen basieren auf Fortschritten in diversen Gebieten, unter anderem bei Informations- und Kommunikationstechnologien sowie bei Mobilitäts-Konzepten. Diese beiden kombiniert ergeben vernetzte Autos. Solche Fahrzeuge mit Kommunikations- Prozessor- und Speicherressourcen wurden in den letzten Jahren immer populärer. In dieser Dissertation wird das Potenzial von vernetzten Autos im Kontext von Smart Cities untersucht. Sie können zum einen genutzt werden, um Effizienz- und Infotainment-Services für Fahrer und Nicht-Fahrer anzubieten. Zum anderen, können diese Fahrzeuge die Kommunikationsinfrastruktur ersetzen bzw. ergänzen, d.h. *virtuelle Infrastruktur* anbieten. Eine Grundidee hierbei ist es, vernetzte Autos zu gruppieren und dabei sogenannte Vehicular Clouds zu bilden. Dabei behandeln wir Forschungsfragen zu Cloud Access, Suche und Nutzung von Services sowie zur Verwendung von Ressourcen. Zuerst untersuchen wir vernetzte Autos als Kernelement von Smart Cities. Hierfür präsentieren wir eine Architektur, um die Suche und Nutzung von Services in Städten zu ermöglichen. Die Ergebnisse zeigen, dass die vorgestellten Ansätze auch bei einer geringen Anzahl von vernetzten Autos gut funktionieren. Darüber hinaus adaptieren wir das Konzept, um auch ländliche Gebiete und Autobahnen zu unterstützen und dabei Datenverbindungen über große Distanzen zu ermöglichen. Zudem zeigen wir, wie vernetzte Autos für eine virtuelle Netzwerkinfrastruktur genutzt werden können. Dies umfasst Clouds entlang von Straßen und auf Parkplätzen, um eine dauerhafte Verbindung zu vorbeifahrenden Autos zu ermöglichen. Zuletzt untersuchen wir Clouds welche mittels fahrenden Autos gebildet werden und wie sie ihre Daten organisieren. Neben den Algorithmen für die Bildung dieser Clouds, beschreiben wir zwei zentrale Services, um Ressourcen aus dem Internet in die lokale Cloud auszulagern bzw., um aggregierte Daten zu sammeln. Wir diskutieren mehrere Verbesserungen für diese Services, um den Ressourcenverbrauch zu reduzieren und deren Leistung zu verbessern. Zusammenfassend zeigen die vorgestellten Architekturen, dass Ressourcen von

vernetzten Autos gut geeignet sind, um Benutzer bei der Suche und Nutzung von Services zu unterstützen. Darüber hinaus sind besonders parkende Fahrzeuge durch die Nutzung ihrer Kommunikationsmöglichkeiten eine Alternative zum Aufbau neuer Infrastrukturen.

# Contents

# Chapter 1

# Introduction and Motivation

SMART CITIES are a major research area spanning multiple disciplines. Beside fields related to Information and Communication Technologies (ICTs), these also include economics and social sciences. The interest in smart cities is driven both by the rapid urbanization around the world and developments in ICT.

According to the United Nations, in 1950, only 30 % (0.75 billion) of the world's population lived in cities [1]. Today this number is as high as 55 % (4.22 billion) and is expected to reach 68 % until 2050. The growth in population leads to economic, social, administrative, and environmental challenges [2]. Recent years also brought many ICT advancements, including *big data*, *cloud computing*, *smart phones*, and the *internet of Things*. These developments can be used to tackle the challenges raised by urbanization and in the process make the cities *smarter*. But what exactly does that mean and what defines a smart city?

Due to the vast number of different angles to look at these smart cities, there is no clear definition what a smart city actually is [2]–[6]. According to Rong et al. [3], the keyword *smart* can differ heavily in scope, be it a simple process improving the quality of life for selected citizens or restructuring a complete governmental process. Over the years, many surveys tried to clarify the concept [2]–[5] or provided their own definitions [7]–[9]. For example, Albino, Berardi, and Dangelico [4] summarize the most popular characteristics used to define a smart city as

- the cities networked infrastructure for efficiency and cultural development,
- urban development and related activities,
- inclusion of residents and social capital,
- and the natural environment as a future resource.

These characteristics formulated as goals can be found at the top of Figure 1.1.

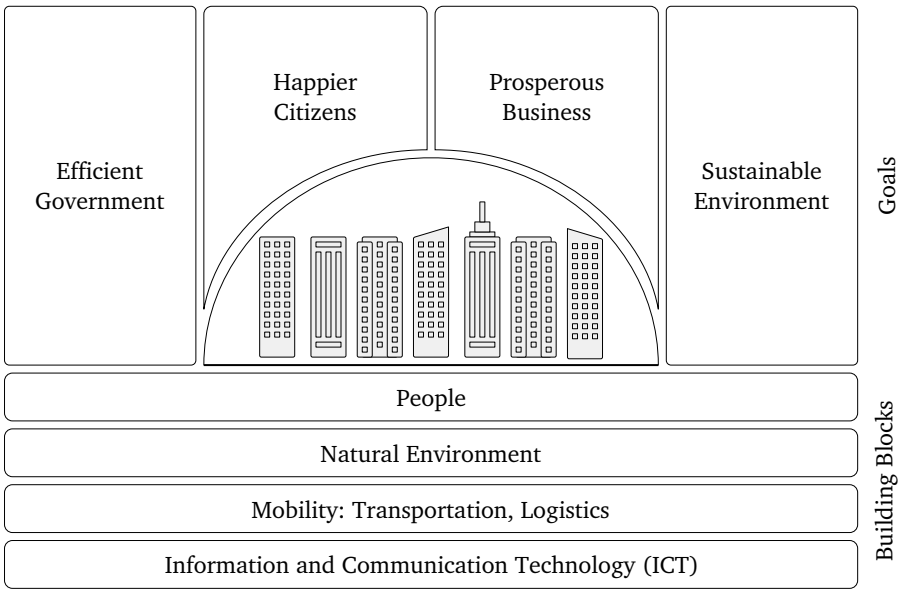


**Figure 1.1** – The building blocks of a smart city and its goals (based on Yin et al. [2] and Albino, Berardi, and Dangelico [4]).

Furthermore, Figure 1.1 shows typical building blocks for smart cites. One of the key smart city enablers are indeed ICTs [2]–[5], [7], [10]. According to Yin et al. [2], the development of today's smart city research is even driven by ICT. In their survey, they concluded that basically all proposed smart city architectures are based on some form of data sensing and collecting. Such city-spanning architectures, like the one proposed by IBM [8], guide the ICT research direction and lead to development of appropriate solutions. Multiple such solutions have been surveyed by Rong et al. [3] who tried to categorize them based on their handling of data. More than 70 % of the architectures focused on data acquisition, close to 60 % on data transmission, and again 70 % on data storage. Similarly, 70 % of the surveyed concepts discuss the processing of data and the integration of various applications. Sensors can be considered a cheap solution to collect a wide range of data, e.g., noise, pollution, temperature, or multimedia. Overall, these numbers indicate that generating, collecting, and processing of data is a core characteristic of a smart city.

Another enabler for smart cities is (*smart*) *mobility*. Seemingly not considered to be as significant as ICTs by various surveys [4], [5], [10], it can nevertheless be an important building block [2], [7], [11]. Useful mobility concepts are able to tackle many issues related to smart cities, including improving public transportation and logistics, or reducing pollution. Benevolo, Dameri, and D'Auria [11] list the core mobility objectives for a smart city. Among them are a reduction of pollution, an improved traffic flow, an increase in people's safety, and reduced transfer costs.

Several of these objectives can be achieved by using vehicles together with ICT systems. One example would be to optimize the traffic by collecting data about vehicular densities throughout the day. This information is then aggregated and processed to improve the traffic flow. One example of such an application are Traffic Information Systems (TISs) [12]. By exchanging and collecting data from vehicles, such a system should be able to provide improvements to drivers, including reduced fuel consumption and travel times, and improving the overall flow of traffic. Another example of an ICT system are Virtual Traffic Lights (VTLs) [13]. Hereby, vehicles cooperatively select the next traffic light phase to avoid long waiting times. This is done without any additional infrastructure using only wireless communication capabilities of vehicles. Beside improving the traffic flow and reducing travel times for individual drivers, VTLs are able to reduce the amount of $CO_2$ emitted by cars [14]. All in all, with concepts for smart mobility, it is possible to improve citizens live in a smart city in numerous ways.

## 1.1 Connected Cars

A combination of both these building blocks, ICTs and smart mobility, are *connected cars*. Due to their mobility, they will be ubiquitous in future smart cites and, hence the name, come equipped with communication capabilities. In this thesis, we consider connected cars to be vehicles equipped with wireless communication capabilities. The communication technology of choice can either be cellular, like LTE, or Dedicated Short-Range Communication (DSRC), e.g., built upon the vehicular networking standard IEEE 802.11p. These technologies are used to communicate with infrastructure (V2I) and with other cars (V2V). Right now, many of the more expensive cars already come equipped with LTE and use it to provide services to drivers, passengers, and the car manufacturers themselves. Similarly, connected cars with DSRC capabilities are not a concept from the future, but are already available. Whether you consider Japan (*ITS Connect* from Toyota [15]), the U.S. (General Motors [16]), or Europe (*WLANp* from Volkswagen [17]), it is possible to buy cars equipped with technology based on IEEE 802.11p. The current strategy for DSRC, for example by Volkswagen, is to sense the local surroundings of a car to provide information to the driver [17]. Furthermore, they intend using the technology to form convoys of networked trucks, so called platoons. The goal of these platoons is to reduce accidents, fuel consumption, and in turn $CO_2$ emissions [18]. Even with connected cars slowly becoming available, research in the area of vehicular networking is still very much active. This is partly because connected cars do not only provide transportation and communication capabilities, but also additional

resources, namely processing power and storage space. This combination enables connected cars to provide more services in future (and even current) smart cities.

Services provided by connected cars can be for drivers and non-drivers alike[1]. Usually, they are grouped in various categories [19]–[21]. All of these classifications include a *safety application* category, which is currently one of the main selling points for putting connected cars on the road [15]–[17]. The main goal of these applications is to reduce the probability of damage to vehicles and persons [21]. Examples are intersection collision warning systems [22], [23], head on and rear-end collision warnings [22], [24], or emergency vehicle services [22], [25]. One already rolled out system is *eCall* [26], mandatory for all new cars in the European Union since April 2018. In case of an accident, *eCall* uses cellular networks to send information like position and driving direction to the emergency services. The goal of *eCall* is to reduce response time for such accidents. In its way of handling data, the system shows the characteristics of a smart city service: collecting data using sensors and processing it using ICTs. Another application category are non-safety services, often further distinguished between *efficiency* and *infotainment*. The goal of efficiency applications, amongst other things, is to coordinate and, in turn, improve the flow of traffic [21]. Examples include TISs [12], [27], [28] or automatic toll collection systems [29]. The final category, infotainment, includes multimedia streaming [30], map and Point Of Interest (POI) updates [31], or access to internet services.

Beside developing these services, there are approaches to bring concepts from other research areas to vehicular networks. In the scope of this thesis, we discuss

[1]Please note that, in this thesis, we use the terms service and application generally interchangeable.

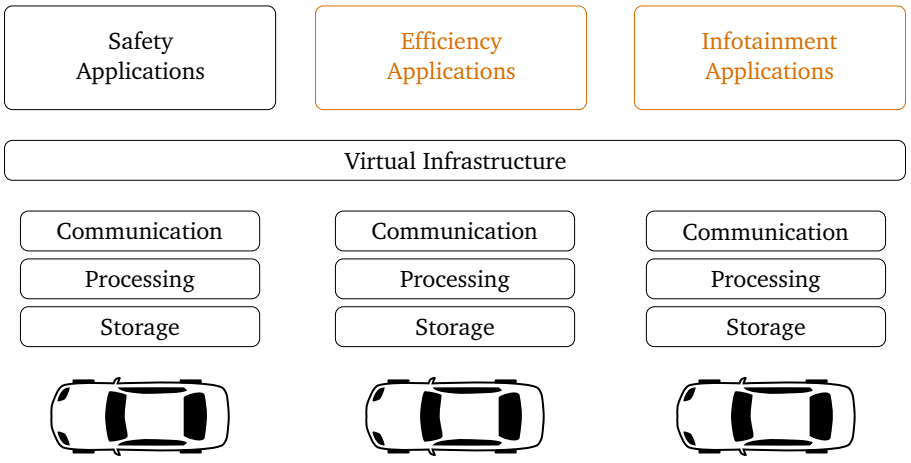


**Figure 1.2** – The role of virtual infrastructure where connected cars work together to provide infrastructure on top of which applications are running. Highlighted are the categories this thesis focuses on.

architectures enabling (additional) services based on the idea of connected cars working together. One example of such an approach is the Vehicular Cloud (VC). Its goal is to group cars in order to provide their resources for offloading tasks from the internet [32], [33]. This is especially useful for services relying primarily on local data. For such applications, there is usually no need to upload data to a data center or to the internet. This would only induce unnecessary delays and in turn leads to poor application performance for the end user when the data is requested later. As examples, Gerla [32] discusses urban surveillance and vehicular traffic management applications. Similar, Mobile Edge Computing (MEC) attracts interest from vehicular network researchers [34], [35]. Here, the idea is to use resources at the edge of the network to perform computational tasks or store data in order to avoid uploading data. If a user connects to a car, it is usually at the edge of the network, hence connected cars are a perfect MEC enabler.

Both these approaches, VC and MEC, are combined in the Micro-Macro Cloud (MMC) architecture by Higuchi et al. [36]. Its idea is to hierarchically group cars in smaller (micro) and large (macro) clouds and in turn makes them better organizable. Usually, macro clouds are city-spanning clusters consisting of multiple micro clouds. Together, these clouds are envisioned to offload resources (MEC) and to provide services (VC). While the concept itself has been established, there are still multiple open issues regarding the architectures components. All of these approaches can be summarized as another application category: *Virtual Infrastructure*. The role of virtual infrastructure is visualized in Figure 1.2. It shows cars using their resources to form virtual infrastructure on top of which the previously discussed applications are running.

In this thesis we propose architectures enabling connected cars to provide services by bringing the required resources closer to users. These architectures are based on the idea of cars working together, essentially forming *virtual infrastructure*. We focus on urban environments, reflecting future smart cities and outline architectures for parked (Section 4.3) and driving cars (Chapter 3 and Section 4.4) alike. These architectures need to overcome several limitations like connectivity between moving cars, varying communication conditions due to wireless effects, or limited availability of connected cars during their introduction phase.

## 1.2 Challenges

There are many challenges when it comes to smart cities in general, and for utilizing connected cars in particular. One key issue for future smart cities is the handling of large amounts of data, especially regarding collection and transmission. This is best seen in the push for the next generation of cellular communication technologies, i.e., 5G [37]–[39]. Requirements mentioned are for example lower latency, higher data rates, network scalability, and longer battery lifetime. But, even if the appropriate infrastructure exists, it does not necessarily imply that the outlined requirements can be fulfilled. To begin with, building new infrastructure costs money and it is not yet clear how fast 5G will be adopted. For example, LTE coverage in the European Union is now close to 100 % [40][2]. Therefore, telecommunications operators might be inclined to upgrade only certain profitable areas to 5G. Furthermore, coverage does not imply a working connection. This was shown in a report of the *simTD* project [41], where a user was driving around Frankfurt am Main, Germany and tried to use the cellular network to connect to a server. Roughly 8 % of the connections failed, which is far too high for providing a good user experience for many applications. As connected cars will become ubiquitous, they are a (potentially cheaper) alternative to building new communication infrastructure.

If we use connected cars to provide services building upon virtual infrastructure, new challenges arise. Among others, which algorithms to use in order to organize cars, how to identify services, and how to connect them. Some of these questions have been answered in the context of Mobile Ad Hoc Networks (MANETs), where many nodes are organized using clustering concepts. Nevertheless, many of these concepts are geared towards saving energy, which is not an issue in vehicular networks [42]. Similarly, these algorithms work with mobility patterns common in MANETs, which are different from cars. To make this even more difficult, car movement varies with environment, i.e., if they are on a freeway, in a city, or if they are parked. Therefore, either new approaches are needed or existing ones need to be adapted to be sufficient [43]. Over time, clustering concepts for vehicular networks have been developed [44]. But many of them are geared towards a single application type and do not consider other services running on the same car. Similarly, these algorithms often focus on a single environment, e.g., freeway or urban (cf. Section 4.2.3). In this thesis, we try to focus on investigating the load on the wireless channel in order to verify if running our solutions leaves space for other applications and services.

The briefly introduced MMC architecture [36] tries to solve challenges in the context of virtual infrastructure. MMC outlines a concept to hierarchically organize clouds, but the large number of involved components and the relations between

[2]The report defines coverage as availability to households. Therefore, areas without anyone living there are not included.

them raises new questions. In the scope of this thesis, we focus on those related to micro clouds and their services. While there exist clustering solutions for vehicular networks [44], it is not clear how well suited they are for micro clouds. Another challenge is the management of data for both offloading and service provision. Finally, if there is a way to form micro clouds, the next step is to determine suitable locations for them. This might depend on numerous things, e.g., connected car density, connectivity to macro clouds, and available resources. The last challenge has been recently investigated by Higuchi, Dressler, and Altintas [45] who proposed an algorithm for selecting micro cloud locations based on consistent resource availability.

Another question which arises, is the technology to use. Currently there are multiple competitors trying to convince car-makers to rely on their approach. This includes LTE (plain cellular or advanced approaches like V-C2X [46]), Wireless LAN (WLAN) (Wi-Fi or specific vehicular networking standards based on IEEE 802.11p [47]), or even other ideas like Visible-Light Communication (VLC) [48]. All of these technologies have in common that they have different strengths and weaknesses. For example, consider LTE in the form of a cellular network. Any message sent from one car to its neighbors has to be routed through the backbone (or at least via the base station) incurring additional and unnecessary delay. Using an ad-hoc technology based on IEEE 802.11p would transmit the message potentially faster. But, if there is the need to communicate with cars farther away than a few hops, IEEE 802.11p would be a bad choice and LTE would probably deliver better results. Therefore, heterogeneous networking could be a solution [20]. Hereby, a connected car is equipped with multiple communication modules, e.g., LTE and WLAN. In such a scenario, the car would be able to switch between technologies depending on the one best suited for the current application.

While at some point in the future, connected cars will probably be ubiquitous, this is not the case right now. Over time, the penetration will slowly grow, but services relying on a high fraction of equipped cars will probably not work initially. Therefore, a system's performance under such conditions should be investigated and, if necessary, alternative or assisting approaches be considered.

To summarize, the key challenges are (1) how to organize connected cars, (2) how to provide virtual network infrastructure, (3) how to form and utilize micro clouds, (4) how to handle a low penetration rate, and (5) what technologies to use for interconnecting all of this. In this thesis we focus on the first four challenges. The fifth challenge regarding technology is not a core topic, but is considered throughout the thesis when presenting the architectures. Similarly, the fourth challenge relating to low penetration rates is investigated as a parameter in all proposed architectures. Generally, all these different challenges lead to more fine-grained research questions.

## 1.3 Research Questions

The core goal of this thesis can be summarized as developing architectures enabling services (primarily of the efficiency and infotainment categories) using vehicular networks. Looking back on the previously mentioned challenges, we focus on three core topics, each with its own research questions:

- **Information Hub for Smart Cities**: First, we are interested in how to build an architecture enabling users to exploit connected cars to discover and utilize appropriate services. This raises the following questions: Who are the involved actors? Which access procedures are required for users to connect to the network? How are provided services identified and how to search for them? Considering this is done in an urban environment, how can we extend such an architecture to make it work in rural/freeway environments?

- **Virtual Network Infrastructure**: Second, we want to exploit parked cars in order to provide virtual infrastructure. The core question is now how to build such a system in order to make the cars successfully work together. If many cars in a parking lot are active, this could lead to unnecessary load on the wireless channel due to the vast number of control messages. Therefore, we are asking how to reduce the load on the wireless channel. In order to transfer large amounts of data, it is necessary to establish longer connections to such a virtual infrastructure node — how can we accomplish this?

- **Micro Clouds and their Data**: Third, we want to design micro clouds using moving cars. Hereby, the focus is on two things: using clustering to form micro clouds and data management. Beside the question of how to form micro clouds, we are interested which core services providing data are available on top of which other, user-facing, applications can be built. This directly leads to the question how such services perform in our micro clouds and how we are able to improve this performance.

## 1.4 Our Contributions

Based on the outlined research questions, we now summarize the contributions presented this thesis. First, regarding building an *Information Hubs for Smart Cities*, we present an architecture named *Car4ICT*, which uses connected cars to provide services for users. We outline the three involved actors, how users are able to access the network, and how to offer, discover, and utilize services. Connected cars are at the core of this architecture and support users throughout the process. To enable

Car4ICT, we propose our concept of service identification and how it can be used both to describe a service, and to search for a range of services. Finally, to discover services not only in cities, but also in rural areas or along freeways, we present the changes required to make the Car4ICT architecture work in such environments.

Second, as an answer to the research questions related to *Virtual Network Infrastructure*, we show how we envision parked cars to form micro clouds and provide such a virtual network infrastructure. Users are not able to connect to all involved parked cars, but only to a subset of them, so called *gateways*. We outline how to select appropriate gateways and how a user can perform handovers to connect to subsequent gateways while driving by. This enables any user to connect longer to the virtual infrastructure and use services requiring long-lasting data transfers. In our evaluation, where we use Car4ICT as an application running on top, we show that gateway selection significantly reduces the load on the wireless channel while nearly maintaining service performance.

Third, to answer questions related to *Micro Clouds and their Data*, we outline an architecture for stationary micro clouds using moving cars. Besides building the system using an algorithm called *geographic clustering*, we present two core data management services for micro clouds. These should be seen as a base for user-facing applications. One of them collects data from the cars participating in the micro cloud, while the other preserves data locally, essentially offloading macro clouds. We evaluate these services in the context of the proposed micro clouds and discuss various performance improvements for them.

## 1.5 Thesis Structure

Following this introductory chapter, *Chapter 2* discusses the fundamentals and the state-of-the-art in related areas. First, we present literature discussing smart city concepts. Second, we show how connected cars can become an important building block in such a smart city and what are the enabling technologies. Third, we focus even more on these connected cars by discussing existing and envisioned services and applications for them. As a core example of how to provide virtual infrastructure, we discuss the MMC architecture, which incorporates multiple approaches, namely VC and MEC. Fourth, we outline the used methodology, namely vehicular network simulation. As a concrete example, we present the network simulator *Veins LTE*. It integrates two simulation frameworks to simulate heterogeneous vehicular networks based on IEEE 802.11p and LTE.

*Chapter 3* presents the *Car4ICT* architecture, which enables users to exploit connected cars for service discovery and utilization. In this chapter, we focus on the research questions regarding how connected cars can be an information hub

of a smart city. We present the Car4ICT architecture from multiple angles. First, we outline the involved actors and how users can access Car4ICT to search for and utilize services. This is followed by our service identification approach called *identifiers*. Hereby, we present how these identifiers can both be used to describe services and how to search for them. The last part of the architecture discussion is how users provide and consume services after they have been discovered. All these parts of Car4ICT are then evaluated in an urban Manhattan grid scenario. Finally, the chapter adapts the presented concepts to use them in rural areas. Therefore, we present an adapted version of Car4ICT with a focus on long-distance connections. In the process we evaluate the Contention Based Forwarding (CBF) algorithm and its parametrization. By doing that, we discover that the default parameters proposed by the European Telecommunications Standards Institute (ETSI) can be improved.

In *Chapter 4* we present two different kinds of micro clouds. Both are based on stationary clusters of connected cars, one of them relying on parked cars, the other on moving cars. These micro clouds are part of the MMC concept, providing services to users and offloading resources. The first proposed micro cloud is formed in a parking lot. Its intention is to provide additional virtual networking infrastructure without incurring costs by exploiting parked cars. First, we present the necessary clustering algorithm and its requirements. The remaining part focuses on improving the performance of the micro cloud. In order to reduce the load on the wireless channel, we outline a system to select a subset of the parked cars as *gateways*. These cars coordinate the communication between users and the micro cloud. To enable longer lasting connections, and in turn the transfer of larger files, we also outline our *handover* approach where a user's connection is passed from one gateway to the next. Our second micro cloud algorithm presents a concept where micro clouds are formed at specific geographic locations. In our case, these are junctions where cars close to the junction are put into one cluster. Again, we start by outlining the underlying clustering algorithm and what we consider the bare minimum needed in order to employ it. Afterwards, we outline two basic micro cloud services for collecting and preserving data. Both these services are envisioned to be used as a base for other, potentially user-facing, applications. Finally, we evaluate our micro clouds using these services and discuss potential improvements for them.

Lastly, in *Chapter 5* we discuss which of the research questions we were able to answer. Moreover, while answering these questions, new issues and ideas arose. We shortly discuss those and point towards potential future research directions.

## Publications

In this section, I outline the publications I was involved in while working on my Ph.D. It includes both the papers this thesis is built on and additional publications published throughout the years.

### Publications This Thesis is Built On

1. F. Hagenauer, F. Dressler, and C. Sommer, "A Simulator for Heterogeneous Vehicular Networks," in *6th IEEE Vehicular Networking Conference (VNC 2014), Poster Session*, Paderborn, Germany: IEEE, Dec. 2014, pp. 185–186
   My contributions to this conference paper were the integration of the cellular network simulator *SimuLTE* into the *Veins* simulator for vehicular network simulation in the form of a new simulation framework, *Veins LTE*. Furthermore, I evaluated the new framework by performing tests using a clustering algorithm built upon heterogeneous vehicular networks.

2. O. Altintas, F. Dressler, F. Hagenauer, M. Matsumoto, M. Sepulcre, and C. Sommer, "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)*, Hong Kong, China: IEEE, Apr. 2015, pp. 654–659
   My contributions to this conference paper were the design, implementation, and evaluation of the proposed Car4ICT architecture. I was involved in developing the architecture based on the core premise: Cars being a service provider in future ICT systems. Based on potential use cases, I categorized the different Car4ICT entities and established a way to identify services using concepts from the area of Named Data Networking (NDN) and Information-Centric Networking (ICN). Finally, I implemented the architecture in our simulation environment and conducted evaluations with a focus on service discovery.

3. F. Hagenauer, C. Sommer, R. Onishi, M. Wilhelm, F. Dressler, and O. Altintas, "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)*, San Francisco, CA: IEEE, Apr. 2016, pp. 788–793
   My contributions to this conference publication were the extension of the Car4ICT architecture for rural areas and freeways. I evaluated what conceptual changes are required in order to connect distant users with each other. As a result, the service discovery mechanisms and its underlying routing had to be changed. Furthermore, I focused on the parameterization of the CBF

routing algorithm. For this, I created a realistic freeway scenario based on real world map and traffic data. This was then used to conduct a parameter study as well as an investigation into the effects of the penetration rate of equipped vehicles onto the performance.

4. F. Hagenauer, F. Dressler, O. Altintas, and C. Sommer, "Cars as a Main ICT Resource of Smart Cities," in *Smart Cities and Homes - Key Enabling Technologies*, M. S. Obaidat and P. Nicopolitidis, Eds., Elsevier, May 2016, pp. 131–147
   My contributions to this book chapter were the extension of the Car4ICT architecture with additional features. These updates included better service comparison and improved differentiation of the various entities. Finally, I evaluated these extensions by adapting the existing implementation of the Car4ICT modules.

5. F. Hagenauer, C. Sommer, T. Higuchi, O. Altintas, and F. Dressler, "Using Clusters of Parked Cars as Virtual Vehicular Network Infrastructure," in *8th IEEE Vehicular Networking Conference (VNC 2016), Poster Session*, Columbus, OH: IEEE, Dec. 2016, pp. 126–127
   My contributions to this conference paper were taking part in the development of the proposed virtual network infrastructure. To make the concept a reality, I participated in the preparation of the outlined central research questions.

6. F. Hagenauer, C. Sommer, T. Higuchi, O. Altintas, and F. Dressler, "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)*, Snowbird, UT: ACM, Oct. 2017, pp. 57–64
   My contributions to this conference paper were the design, development, and evaluation of the proposed micro cloud algorithm. Based on the idea that parked cars can be used as virtual network infrastructure, I developed a solution using the Virtual Cord Protocol. Hereby, I focused on a core part of the architecture, namely gateway selection. Such a selection lowers the number of nodes a passing car can connect to. This selection barely reduces the performance of applications exploiting the virtual infrastructure. Finally, I evaluated the architecture in varying circumstances with a focus on the influence of the gateway selection both in an artificial and a realistic scenario.

7. F. Hagenauer, C. Sommer, T. Higuchi, O. Altintas, and F. Dressler, "Vehicular Micro Cloud in Action: On Gateway Selection and Gateway Handovers," *Elsevier Ad Hoc Networks*, vol. 78, pp. 73–83, Sep. 2018
   My contributions to this journal article were the improvement and evaluation

of the extended virtual network infrastructure algorithm. I focused on developing the handover part of the algorithm enabling longer connectivity. By performing handover, cars can transfer more data or use complex protocols. Eventually, I evaluated the algorithm in two realistic scenarios and focused on the performance of the handover improvement.

8. F. Hagenauer, C. Sommer, T. Higuchi, O. Altintas, and F. 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)*, Snowbird, UT: ACM, Oct. 2017, pp. 31–35
   My contributions to this conference paper were the design and evaluation of an algorithm forming micro clouds at geographic relevant locations, i.e., a map-based approach. I developed the idea where micro clouds are formed using clustering concepts at specific locations coordinated by Access Points. The concept was evaluated in a basic artificial scenario with a single application, data collection from Cluster Members. Furthermore, the evaluation was not only done using IEEE 802.11p, but also in combination with LTE.

9. F. Hagenauer, T. Higuchi, O. Altintas, and F. Dressler, "Efficient Data Handling in Vehicular Micro Clouds," *Elsevier Ad Hoc Networks*, vol. 91, p. 101 871, Aug. 2019
   My contributions to this journal article were the development, extension, and evaluation of the previously proposed geographic clustering algorithm. In addition to the data collection service, I took part in the development of a further service, which keeps local data in the micro cloud. For this service, I developed improvements increasing the performance significantly. Finally, I evaluated these findings in a realistic Manhattan grid scenario for varying traffic densities showing how the improvements affect the performance.

## Additional Publications

1. F. Hagenauer, P. Baldemaier, F. Dressler, and C. Sommer, "Advanced Leader Election for Virtual Traffic Lights," *ZTE Communications, Special Issue on VANET*, vol. 12, no. 1, pp. 11–16, Mar. 2014

2. F. Hagenauer, C. Sommer, S. Merschjohann, T. Higuchi, F. Dressler, and O. Altintas, "Cars as the Base for Service Discovery and Provision in Highly Dynamic Networks," in *35th IEEE Conference on Computer Communications (INFOCOM 2016), Demo Session*, San Francisco, CA: IEEE, Apr. 2016, pp. 358–359

3. A. Memedi, F. Hagenauer, F. Dressler, and C. Sommer, “Cluster-based Transmit Power Control in Heterogeneous Vehicular Networks,” in *7th IEEE Vehicular Networking Conference (VNC 2015)*, Kyoto, Japan: IEEE, Dec. 2015, pp. 60–63

4. M. Mutschlechner, F. Klingler, F. Erlacher, F. Hagenauer, M. Kiessling, and F. Dressler, “Reliable Communication using Erasure Codes for Monitoring Bats in the Wild,” in *33rd IEEE Conference on Computer Communications (INFOCOM 2014), Student Activities*, Toronto, Canada: IEEE, Apr. 2014, pp. 189–190

5. C. Sommer, F. Hagenauer, and F. Dressler, “A Networking Perspective on Self-Organizing Intersection Management,” in *IEEE World Forum on Internet of Things (WF-IoT 2014)*, Seoul: IEEE, Mar. 2014, pp. 230–234

6. F. Dressler, G. S. Pannu, F. Hagenauer, M. Gerla, T. Higuchi, and O. Altintas, “Virtual Edge Computing Using Vehicular Micro Clouds,” in *IEEE International Conference on Computing, Networking and Communications (ICNC 2019)*, Honolulu, HI: IEEE, Feb. 2019

7. C. Sommer, D. Eckhoff, A. Brummer, D. S. Buse, F. Hagenauer, S. Joerer, and M. Segata, “Veins – the open source vehicular network simulation framework,” in *Recent Advances in Network Simulation*, A. Virdis and M. Kirsche, Eds., to appear, Springer, 2019, pp. 1–1

# Chapter 2

# Fundamentals

In this chapter, we present the fundamentals and the work related to the research presented in this thesis. We start the chapter with a literature review about what makes a smart city actually *smart* and what definitions have been coined over the years. Hereby, we outline common characteristics with a focus on Information and Communication Technology (ICT) and mobility. This is followed by a discussion of connected cars and what technologies they use to communicate. Furthermore, we present services and applications for connected cars. As a special use case, we discuss *Virtual Infrastructure*. In this concept, connected cars provide infrastructure for various services, e.g., by offloading resources or collecting local data. The final section of this chapter presents our most widely employed evaluation methodology: simulation. In the process, we present *Veins LTE*, a simulation framework for heterogeneous vehicular networks supporting both Dedicated Short-Range Communication (DSRC) and cellular communication.

Parts related to *Veins LTE* (Section 2.5.2) are based on the following publication:

- F. Hagenauer, F. Dressler, and C. Sommer, "A Simulator for Heterogeneous Vehicular Networks," in *6th IEEE Vehicular Networking Conference (VNC 2014), Poster Session*, Paderborn, Germany: IEEE, Dec. 2014, pp. 185–186
  My contributions to this conference paper were the integration of the cellular network simulator *SimuLTE* into the *Veins* simulator for vehicular network simulation in the form of a new simulation framework, *Veins LTE*. Furthermore, I evaluated the new framework by performing tests using a clustering algorithm built upon heterogeneous vehicular networks.

## 2.1 Smart Cities

Architectures presented in this thesis are envisioned to be used in smart cities. According to Angelidou [65], early concepts of future cities emerged starting from the 1850s. But, in the 1960s, with the emerge of new technologies, the idea reached a broader public. Initially, the presented ideas covered the mechanization of cities, but quickly focused on the newly available information and its processing in an urban context [65].

But, what exactly makes a city smart? While widely used in literature, there is seemingly no single definition of the term *smart city*. One explanation for this, is the large number of involved scientific fields, e.g., ICTs, economics, and social sciences. Depending on the particular research interest, there might be different things constituting a smart city [9], [10]. In the following, we discuss publications investigating smart city definitions with a focus on what, according to the authors, constitutes a smart city. Some of these characteristics can be put into one of two categories: *goals* and *building blocks*. Goals are what should be achieved with a

smart city, e.g., improve quality of life, reduce traffic flow, or make money. Building blocks are the essentials of a smart city, i.e., what is needed to reach the goals. This might include natural resources, technologies, or humans.

In 2008, Hollands [5] criticized that many cities call themselves smart without any meaning behind it. Consequently, he surveyed what features these cities used to call themselves smart. According to his analysis, there is a wide range of characteristics involved, e.g., information technologies, business innovation, governance, communities, and sustainability. Furthermore, the author points out that networked infrastructure is a key building block spanning transport, business services, and housing. Caragliu, Del Bo, and Nijkamp [6] discuss smart cities in Europe and presents six core characteristics regularly found in literature. Based on their research, they come up with the following smart city definition:

> " We believe a city to be smart when investments in human and social capital and traditional (transport) and modern (ICT) communication infrastructure fuel sustainable economic growth and a high quality of life, with a wise management of natural resources, through participatory governance. " (quote from [6])

According to Caragliu, Del Bo, and Nijkamp [6], building blocks are various forms of investments, transportation, and ICT. They are used to achieve certain goals, such as economic growth, quality of life, or environmental sustainability.

More recently, extensive analysis of smart city definitions like the ones provided by Yin et al. [2] or Albino, Berardi, and Dangelico [4] try to sketch out the core characteristics of a smart city. After analyzing various sub-domains of smart city research, Yin et al. [2] provide four smart city goals: efficient government, happier citizens, prosperous business, and sustainable environment (cf. Figure 1.1). Like Hollands [5], the authors mention ICT as a core building block. By collecting a list of smart city definitions, Albino, Berardi, and Dangelico [4] try to find similarities leading to a clearer picture what makes a smart city actually smart. According to them, common characteristics are networked infrastructure, business-led urban development, social inclusion, and the natural environment. A different approach was taken by Giffinger and Haindlmaier [7], who developed a categorization of smart cities. Their goal was to rank European cities based on these categories. To achieve this, they built a hierarchically model where 74 indicators were categorized as 31 factors leading to 6 core smart city categories. These are economy, people, governance, mobility, environment, and living[3]. Overall, despite taking different approaches, the resulting smart city definitions and characteristics are similar.

When we consider complete smart city architectures, one popular example was proposed in 2010 by IBM [8]. With a focus on ICT, Harrison et al. [8] built their

[3] In the original publication they have been all prefixed with the term smart.

concept around the three terms *instrumental*, *interconnected*, and *intelligent*. Hereby, *instrumental* refers to different sources generating data, *interconnected* to collection of this data, and *intelligent* to processing and analysis of the data. Rong et al. [3] propose another architecture based on the data collected in a city. Their goal was to deal with the city's problems and provide citizens with a better life by collecting and analyzing data using ICTs. To achieve this, they propose a six-layered architecture, which takes care of collecting, transmitting, storing, and exploiting the gathered data for various kinds of smart applications.

While the discussed literature mentions mobility as a building block [2], [7], [11], ICT is seemingly considered to be more important. Nevertheless, some scientists like Benevolo, Dameri, and D'Auria [11] focus on smart mobility as a core smart city concept. They highlight six objectives where smart mobility is helpful. These objectives are directly reflected in three smart city goals: digital, where ICT systems are used to solve the objectives; green, where smart mobility is used to reduce the impact of mobility on the environment; and knowledge, where data collection enables smart transportation.

In Table 2.1 we summarize publications considering either ICT or mobility as smart city building blocks. As can be seen, ICT is considered by all of them, while mobility only by some. Furthermore, as mentioned earlier, we can observe that the viewpoint makes a difference when reasoning about what is important by looking at the studies conducted by Nam and Pardo [9], [10]. Both studies consider ICT to be a core building block, but only the study focusing on technology considers mobility as well [9]. The other study, with a focus on management and policy, does not see mobility as important [10].

In this thesis, we focus on connected cars as a building block for smart cities. They can be considered a key enabler as they are able to provide two of the discussed building blocks: ICT infrastructure and mobility.

**Table 2.1** – Summary of publications mentioning either ICT or mobility as a smart city building block.

| Publication | ICT | Mobility |
|---|---|---|
| Hollands [5] | ✓ | |
| Giffinger and Haindlmaier [7] | ✓ | ✓ |
| Harrison et al. [8] | ✓ | |
| Caragliu, Del Bo, and Nijkamp [6] | ✓ | |
| Nam and Pardo [10] | ✓ | |
| Nam and Pardo [9] | ✓ | ✓ |
| Rong et al. [3] | ✓ | ✓ |
| Yin et al. [2] | ✓ | ✓ |
| Albino, Berardi, and Dangelico [4] | ✓ | |
| Benevolo, Dameri, and D'Auria [11] | ✓ | ✓ |

## 2.2 Connected Cars

One of the core ideas of this thesis is to use connected cars as key enablers for smart cities. This is based on the assumption that they can provide two building blocks: ICT and mobility. Being equipped with communication technologies, we can exploit their mobility to provide services throughout a city. They are able to connect to existing infrastructure (Vehicle-to-Infrastructure (V2I)) or to other cars (Vehicle-to-Vehicle (V2V)). But, what kinds of technology are used in connected cars?

Currently there are several networking technologies used (and partly envisioned) for vehicular networks, covering both ad-hoc and cellular networks:

- **WLAN Technologies**: When considering ad-hoc networks, WLAN is the technology of choice for connected cars. Hereby, the focus is usually on the IEEE 802.11p standard [47], an amendment to the Wireless LAN (WLAN) standard IEEE 802.11 [66]. IEEE 802.11p outlines how the Physical layer (PHY) and Medium Access Control (MAC) layer should work. Depending on the region of the world, there are numerous standards building upon IEEE 802.11p: Wireless Access in Vehicular Environments (WAVE) in the US [67], ETSI ITS G5 in Europe [68], and Arib T109 in Japan [69]. It is common to use the term Dedicated Short-Range Communication (DSRC) for this kind of technology, although, strictly speaking, the term describes any kind of short-range communication [67].

  While cars being equipped with IEEE 802.11p are only starting to become available (cf. Chapter 1), Wi-Fi is already widely deployed throughout cities. Mobile phones, access points, and laptops enable connectivity between each other and to the internet. Therefore, Wi-Fi, while not optimized for vehicular networks, might be an option during the introductory phase of connected cars.

- **Cellular Technologies (LTE, 5G)**: While, as discussed in Chapter 1, it is possible to buy cars with IEEE 802.11p, more of the currently available cars are equipped with cellular technology, e.g., 3G or 4G. But, due to its nature as a cellular technology, communication usually requires a base station to work (e.g., an eNodeB (eNB) in case of LTE). This might induce large and potentially unnecessary delays for communication between cars in close proximity. Therefore, in this form, cellular networks are not suited for safety applications, which require low latencies. Over time, such issues have been recognized by the relevant standardization entity 3GPP. They started to work on additional LTE specifications enabling direct communication between nodes without the need of an eNB, called LTE D2D or, more formally, Proximity-based services [70], [71]. Moreover, another specification was introduced, specifically targeting vehicular networks based on LTE. This has been launched as part of 3GPP

Release 14 in a first version as Cellular V2X (C-V2X) [46]. By introducing two additional communication modes, it is now possible for LTE nodes to communicate without any base station. This standard will evolve with 3GPP Release 15 where the focus lies, among others, on autonomous driving, platooning, and sensor sharing [72].

- **Other Technologies**: Beside the main contenders, research is conducted on other technologies and their potential in vehicular networks. Visible-Light Communication (VLC) aims to exploit the light emitted by head- and taillights to communicate with other cars. While this clearly needs line-of-sight to work, it has the potential to transmit large amounts of data without an easy way to eavesdrop [48]. Other technologies include Bluetooth [73] and radar [74].

All the mentioned technologies have their advantages and drawbacks. Due to their different properties, there are situations where one technology excels, but others fail to work well. Multiple properties are discussed by Sommer and Dressler [75] and in a similar way by Zheng et al. [20]. First, there are data rate and delay. On the one hand, LTE is able to provide data rates necessary for entertainment applications, DSRC is not. On the other hand, LTE fails when it needs to deliver messages with a delay in the range of 50 ms . . . 100 ms [76]. Second, cost is an argument for DSRC technologies like WLAN or Wi-Fi. Infrastructure for using cellular networks costs money in two ways: (1) when building the necessary base stations and (2) for maintaining the service. Therefore, telecommunication providers usually demand money for using their services. Third, range is an argument in favor of cellular technologies like LTE. DSRC will likely not be able to reach distances farther away than a few kilometers, while this is not an issue for LTE [75]. Integrating infrastructure, like Roadside Units (RSUs) for DSRC would induce costs, removing one property where DSRC has an advantage.

One solution to these issues is to rely on *heterogeneous vehicular networks*. Zheng et al. [20] discuss such approaches in their survey. They define a heterogeneous network as a network equipped with multiple networking technologies, each with their own networking stack. Each of those include their own MAC and PHY layer. Between these two layers and the upper layers, the authors introduce an additional *Heterogeneous Link Layer*. Its goal is to offer a unified interface for applications to access the different networking stacks.

Using more than one network stack has been proposed already in 2005 by Cavalcanti et al. [77]. Their architecture combines WLAN with cellular technologies, 3G at that time. Over time, the concept has been adapted for vehicular networks, leading to *heterogeneous vehicular networks*. One of the early works was done by Chen and Chan [78]. They proposed the *MobTorrent* system to download data from Access Points (APs) using cellular networks as a control channel. Connected cars are

subsequently used to improve the performance by caching data and providing it to others. More recently, others also proposed to combine WLAN and cellular networks in a similar way [79], [80]. The overall goal of these works is to reduce the message delay or even the number of required messages. But, when it comes to evaluating their algorithm, the authors did not use a simulator with a feedback loop between the road traffic simulator and the network simulator. If such a loop is missing, the traffic is not able to react to received messages and vice versa.

Generally, the architecture prototypes designed in this thesis do not rely on a specific communication technology. Most of them were designed with a focus on DSRC, but can be used with cellular networks or profit from them. This is especially useful in the introductory phase of vehicular networking, where not all cars will be equipped with the necessary technology [75]. In our evaluation, we usually use a network stack based on IEEE 802.11p [66] and IEEE 1609.4 [81].

## 2.3 Applications & Services of Connected Cars

Over the years, many applications and services have been envisioned for connected cars. There have been multiple attempts to categorize them [19]–[21]. Basically, all these categorizations include a *safety* application category. Their goal is to reduce the risk of traffic accidents and to avoid injuries [21]. One example are collision avoidance applications [23], [82] where connected cars periodically announce their presence. This enables other cars to react by warning the driver or stop the car, even if the vehicles are not in line-of-sight of each other. More generally, this is enabled by awareness of the surroundings. Not only allows this to detect other vehicles, but also to warn about potential road hazards. A concrete example are Cooperative Awareness Messages (CAMs) as part of the ETSI ITS-G5 standard in Europe [83]. These messages are intended to be sent at high frequencies (above 10 Hz).

Another set of applications can be categorized as *efficiency* applications. Broadly speaking, their goal is to manage the flow of traffic [21]. As an example, Traffic Information Systems (TISs) use GPS position to provide navigation to drivers and include various further resources to update the provided route. Such information can also be based on data collected from connected cars, e.g., based on their position, current speed, or neighboring cars. There are different approaches how to design such a system, either centralized [75] or in a distributed manner [28]. If we consider a centralized TIS [75], data from connected cars is collected at some central entity and processed before being used for routing. In a distributed system, cars exchange the necessary information with each other without the need for a central coordinator. Another application that can be considered to be part of the efficiency category is platooning, with the goals reducing fuel consumption and pollution, and increasing

safety [18], [84]. In a platoon, the leading vehicle dictates speed and direction. Other vehicles, without needing a driver, follow the leader in close proximity. The coordination is envisioned to be done via wireless communication.

Usually the final category are *infotainment* applications. This category includes for example Point Of Interest (POI) notifications, multimedia downloading, and services accessible via the internet [21]. One concrete example is provided by Malandrino et al. [30], who use connected cars to assist in content downloading. They investigate how parked cars can assist driving vehicles in downloading data via APs. Even more challenging is the concept of multiplayer gaming using connected cars as proposed by Tonguz and Boban [85]. Based on data indicating multiple occupants in a journey, the authors argue that connected cars provide a lucrative incentive for gaming. Furthermore, they outline requirements and list challenges needed to be solved.

Not all applications can be categorized using such schemes, but fall into multiple of them. Consider for example Virtual Traffic Lights (VTLs) [13], where connected cars cooperatively decide who is allowed to drive and who needs to stop in front of a traffic light. Such an application can be considered falling into the safety (avoiding accidents) and the efficiency (improving the flow of traffic) category.

In this thesis, our architecture prototypes are mainly built for efficiency and infotainment applications. The intention was to provide a solid base for such kinds of applications to run on top. While safety applications might work as well, we did not build them with the necessary low delays in mind.

## 2.4 Virtual Infrastructure

As discussed in Chapter 1, in the scope of this thesis, we consider another application category: *Virtual Infrastructure*. This assumes that infrastructure is able to improve the performance of a vehicular network. Beside cellular networks, where base stations take the role of coordinator nodes, DSRC also benefits from infrastructure. So called Roadside Units (RSUs) can be used to extend the range of routing algorithms as proposed by Mershad, Artail, and Gerla [86]. Their goal was to mitigate DSRC issues regarding the communication distance (cf. Section 2.2). In order to achieve this, their algorithm exploits the RSU backbone to transfer messages to other cars farther away. Similarly, Lochert et al. [87] discuss placing such RSUs to overcome the issue of low penetration rate in introductory phases. RSUs, as envisioned by them, run the same applications as connected cars and take part in the collection of traffic information. It should be noted that, even if 100 % of the vehicles are equipped with networking technologies this results in a fragmented network, which

highlights the need for a mitigation [88]. Mentioned actions to avoid disconnected networks include Store-Carry-Forward (SCF) and deployment of RSUs.

But, deploying RSUs costs additional money. Therefore, we propose to use connected cars to provide virtual infrastructure. Hereby, we focus especially on parked cars, which are usually completely turned off. If we use their computing and networking resources, they can provide valuable services [30], [89], [90]. Even more generally, connected cars, parked or moving, are able to provide the underlying infrastructure for services. This is reflected by two concepts where connected cars play a major role: Vehicular Cloud (VC) and Mobile Edge Computing (MEC).

The concept of VCs is roughly seven years old [32], [33], [91]. In its basic version, connected cars form groups and together provide services and offer available resources [92]. Considered services include traditional cloud computing like Storage as a Service or Computing as a Service. Furthermore, the mobility of the cars allows for more specialized examples, such as Network as a Service or Cooperation as a Service [93]. This already nicely fits the concept of virtual infrastructure. Early approaches, like the one proposed by Gerla [32], highlight the mobility of VCs. This makes them versatile in their application, but also increases the difficulty of developing appropriate solutions. By providing local services, these clouds can offload tasks from the internet. Similarly, Olariu, Hristov, and Yan [33], highlight the resources cars are able to provide when forming a VC. The idea of VCs does not only consider moving cars, but can also be extended to parked cars [89], [94]. In such a scenario, the resulting cloud can act similarly to a regular cloud.

The concept of MEC originated with the idea of putting computational resources at the edge of the network and, in consequence, closer to the end-user. Not yet called MEC, *cloudlets* can be considered an early example [95]. The goal was to bring the required resources close to users, so they are able to reach them via WLAN.

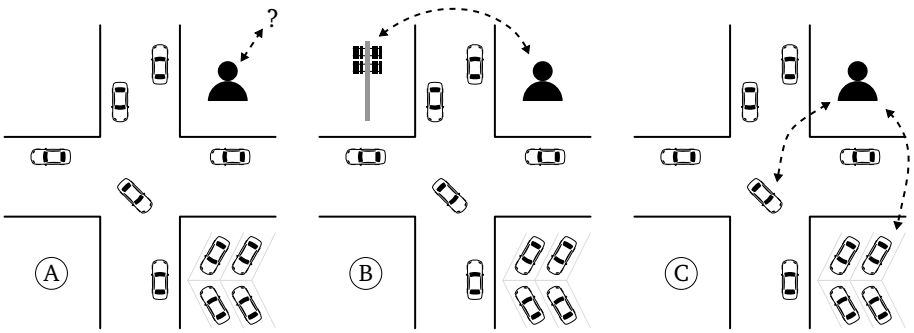


**Figure 2.1** – Example of a user trying to use services. In example A, no service provider is in reach. Example B shows a cellular base station and the user being able to connect to it. Example C shows connected cars, both moving and parked, providing infrastructure.

Similar, *fog computing* tried to move cloud computing from the network's core closer to the edge [96]. Considered services were computation power, storage space, and network communication. More recently, there are approaches to standardize MEC, e.g., by the ETSI [34]. According to them, applications benefiting from resources at the edge of the network include augmented reality, multimedia streaming, and connected cars. More general, Mach and Becvar [35] consider three categories of MEC applications in their survey: consumer oriented, operator and third-party, and network performance and QoE improvement services.

To bring MEC and VC together, Higuchi et al. [36] envisioned the Micro-Macro Cloud (MMC) architecture. Based on a hierarchical organization of clouds, i.e., *micro* and *macro* clouds, their approach has two goals: (1) provide a VC at the edge of the network and (2) work as a base for new applications. Micro clouds are restricted to smaller areas and usually consist only of a few cars. Macro clouds are formed out of multiple micro clouds and can span entire cities. In Figure 2.2, we show an example of this architecture. Micro clouds together form a macro cloud, which itself is then connected to a data center or the internet. Micro clouds can consist of moving

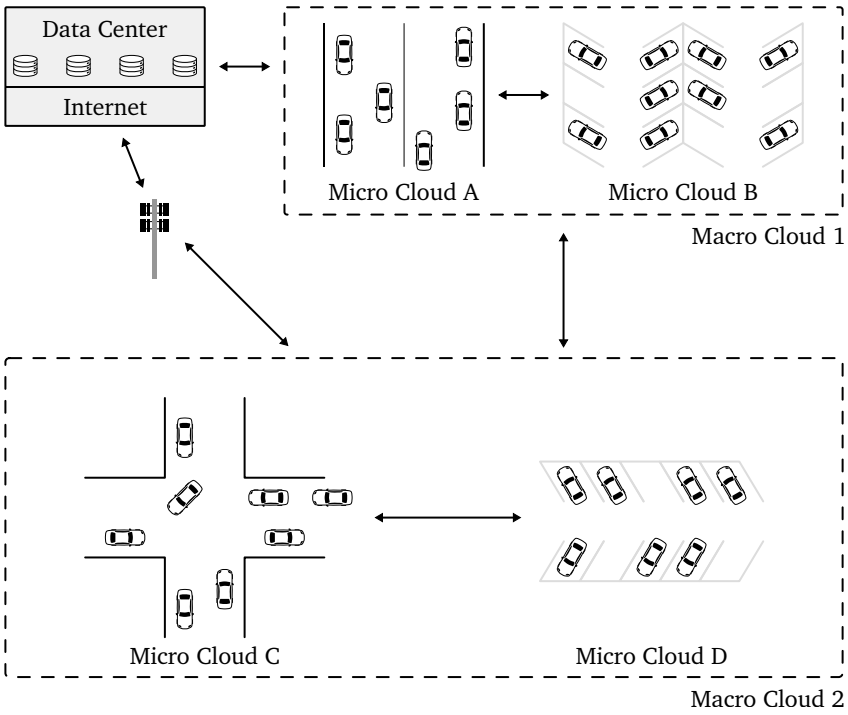


**Figure 2.2** – An overview of the Micro-Macro Cloud (MMC) architecture. It shows 4 micro clouds (A–D) and two macro clouds (1 and 2). Three micro clouds are stationary (B, C, and D), one is mobile (A). Two of the micro clouds consist of moving cars (A and C) and the others of parked cars (B and D). Both macro clouds are connected to a data center or to the internet.

cars (A and C) or of parked cars (B and D). While the concept builds upon existing ideas, MMC itself is a relatively new idea with various open research questions. One of those was investigated by Higuchi, Dressler, and Altintas [45], with a focus on suitable micro cloud locations. Based on the resources available in connected cars, their algorithm determines suitable locations for micro clouds.

The architectures proposed in this thesis nicely fit into the MMC concept. Especially the two clustering approaches presented in Chapter 4 are essentially micro clouds, which are able to provide services as well as offload services from a macro cloud. There, we use parked cars to provide virtual infrastructure, on top of which services are provided to users. Furthermore, we discuss two basic services for such micro clouds which can be seen as the base for additional, user-facing, applications.

## 2.5 Evaluation Methodology

In the remainder of the chapter we discuss our primarily used evaluation methodology, simulation. Usually, there are three approaches how to evaluate the performance of vehicular networks: experimentation, analytical evaluation, and simulation. Experiments are usually quite expensive and are only able to cover a few scenarios and parameter combinations [75]. As our proposed architectures are envisioned to work with numerous participating cars, this was not feasible. Analytical evaluation is generally the preferred methodology, but easily becomes too complex. To tackle this complexity, it is necessary to make assumptions. They tend to oversimplify the system, e.g., by assuming a unit-disk model for communication. Again, this is not feasible for large scale architectures. This leaves us with simulation, which indeed was used as the primary evaluation tool of choice for this thesis.

### 2.5.1 Veins: Vehicular Network Simulation Framework

When using simulation, one needs to be careful to select the correct models and scenarios. Therefore, we relied on a well-established framework, namely *Veins*[4] [98]. Veins couples a network with a road traffic simulator, enabling the evaluation of vehicular networks. It is built using OMNeT++ [99] and initially originated from the MiXiM framework [100]. OMNeT++, at its core, is a discrete event simulator, which has been extended for continuous signals with MiXiM[5]. Veins provides a wide range of well-established models to simulate DSRC communication based on IEEE 802.11p. This includes obstacle shadowing [43], two-ray fading [101], and complete PHY and MAC layers [102]. For road traffic simulation, Veins is tightly

[4] `https://veins.car2x.org/`
[5] note that, current versions of Veins do not rely on MiXiM anymore

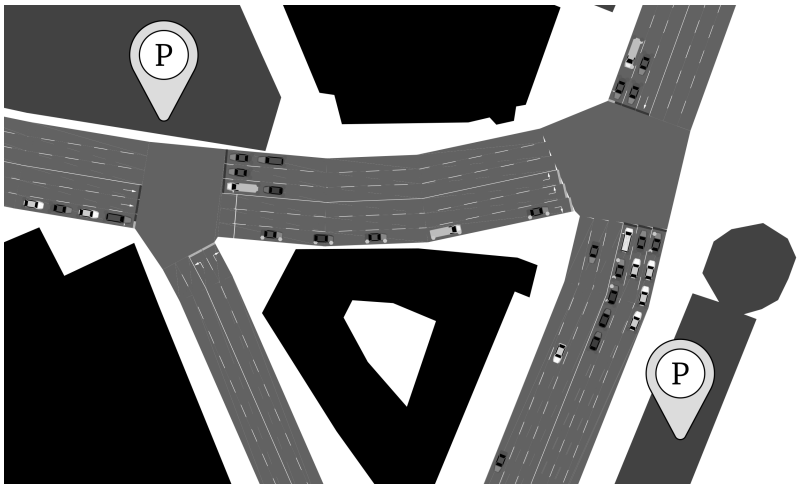


**Figure 2.3** – A screenshot of *SUMO* showing an intersection from the LuST Luxembourg scenario [97]. The markers indicating parking lots were manually added.

coupled to *SUMO*[6] [103]. SUMO provides established mobility models and there exist high-quality scenarios of whole cities, e.g., the LuST Luxembourg scenario [97]. An example of a SUMO simulation can be seen in Figure 2.3.

Over the years, many modules have been developed extending the functionality of Veins. In the following we present one of them, namely *Veins LTE*, which, as the name gives away, extends Veins with LTE capabilities.

### 2.5.2 A Simulator for Heterogeneous Vehicular Networks

In Section 2.2 we briefly outlined why heterogeneous networks are useful in vehicular networks. Beside the already mentioned issues regarding delay and distance, network load can be considered another one. Technologies like IEEE 802.11p, which are based on WLAN, have issues when being operated in areas with a lot of road traffic. This incidentally leads to more load on the wireless channel, which also reduces the performance of vehicular networking applications. Furthermore, the penetration rate of connected cars will be initially very low. Cellular technologies like LTE might not have the infrastructure available to incorporate the additional communicating nodes. And not only using LTE costs money, also the infrastructure necessary for vehicular networking needs to be built [91]. Due to all these issues, heterogeneous vehicular networks can be the solution. By having two network stacks (e.g., LTE and IEEE 802.11p) the one better suited for a task can be used.

In this chapter we present the core components of *Veins LTE*, a simulator enabling the simulation of LTE and IEEE 802.11p as a heterogeneous network stack. It is built

[6] `https://sumo.dlr.de`

by integrating *Veins* [98] with *SimuLTE* [104]. The final product is available as an open source framework.[7]

Note that, at the time of development, *Veins LTE* was one of the first simulators for heterogeneous vehicular networks. Since then, several others were developed, e.g., *Artery* by Riebl et al. [105]. Furthermore, one of the core *Veins LTE* parts, SimuLTE, was extended to support LTE D2D communication [106] and is on track to natively support Veins.

#### 2.5.2.1 Integrating Veins and SimuLTE

*Veins LTE* is based on the previously discussed Veins, which itself is based mainly on OMNeT++. With Veins, modules for road traffic and DSRC simulation were available. The only thing left was an LTE module. This is provided by *SimuLTE* [104]. As both Veins and SimuLTE are based on OMNeT++, it was possible to integrate the cellular functionality.

This integration itself proved to be a tedious task as SimuLTE was not built with the dynamics necessary for vehicular network simulation in mind. In particular, it could not handle nodes being added and removed during runtime, i.e., the simulator expected all nodes to be present at initialization. Moreover, adding or deleting vehicles during runtime resulted in an error. Therefore, our initial task was to modify the LTE stack provided by SimuLTE to be able to handle nodes leaving the simulation and in turn support dynamic vehicular networks. This was done by meticulously going through the *SimuLTE* modules and adding capabilities to avoid crashes and handle messages lost in transmission.

Afterwards, we extended the stack with modules enabling straightforward implementation of new applications. The resulting final stack of *Veins LTE* can be seen in Figure 2.4. At the top, we have an application layer supporting an arbitrary

[7] `http://veins-lte.car2x.org/` and the source at `https://github.com/floxyz/veins-lte`

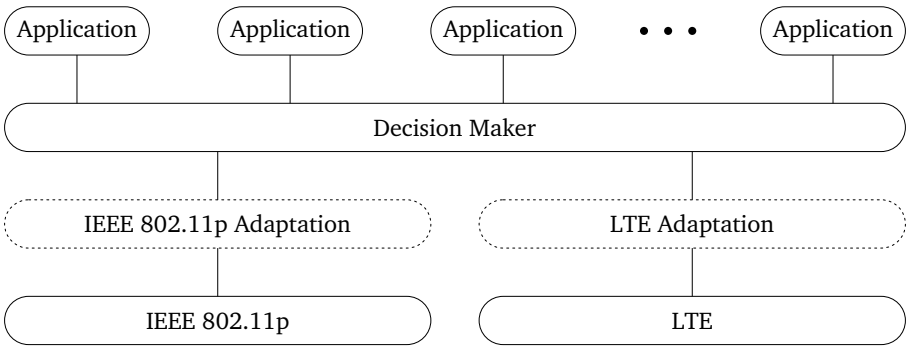


**Figure 2.4** – The *Veins LTE* protocol stack including both the IEEE 802.11p and the LTE functionality; based on [49] © 2014 IEEE.

number of applications. Below it, we can find the *Decision Maker* module. If the application layer does not select a network stack to use when sending messages, the *Decision Maker* can be used to provide this decision. Different algorithms for this decision provide opportunity for interesting future work. If the decision has been made (either by the application or the *Decision Maker*), the message is forwarded to the respective stack. Before being sent, some adaptations are made to add required parameters. This includes protocol headers or simulation framework specific information necessary for a successful message transmission. Afterwards, the message is forwarded to the respective MAC and later NIC modules. For more details about these layers, we refer to the literature outlining them, i.e., Sommer, German, and Dressler [98] and Virdis, Stea, and Nardini [104].

If a message is received by a car, it takes the other way around and moves up the stack. After leaving the network stack, the message is stripped of all technology-dependent information and is forwarded to the application layer.

#### 2.5.2.2 Exemplary Simulation Results

To evaluate that Veins and SimuLTE work as intended, we implemented the clustering outlined by Tung et al. [80] and evaluated it using an artificial Manhattan grid scenario. Using IEEE 802.11p, the algorithm creates clusters, which share information about clusters with an eNB using LTE. This information is then processed and potentially shared with other clusters in the vicinity to avoid collisions.

In Figure 2.5 we show the uplink delay for LTE message sent by the Cluster Head (CH) to the eNB. The delay increases with higher message frequency. Interestingly, the slope decreases between 10 Hz and 20 Hz. The reason for this is an even larger increase in lost packets.

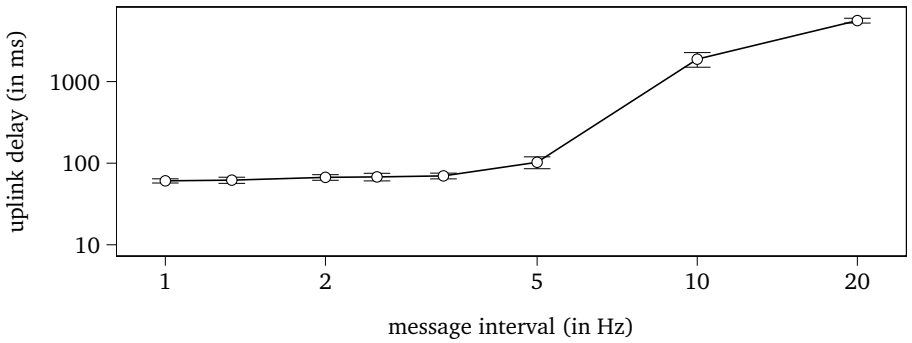


**Figure 2.5** – Exemplary results using the LTE stack of *Veins LTE* with logarithmic scales. Shown are the message interval and the uplink delay for 100 RB with a 20 MHz bandwidth; based on [49] © 2014 IEEE.

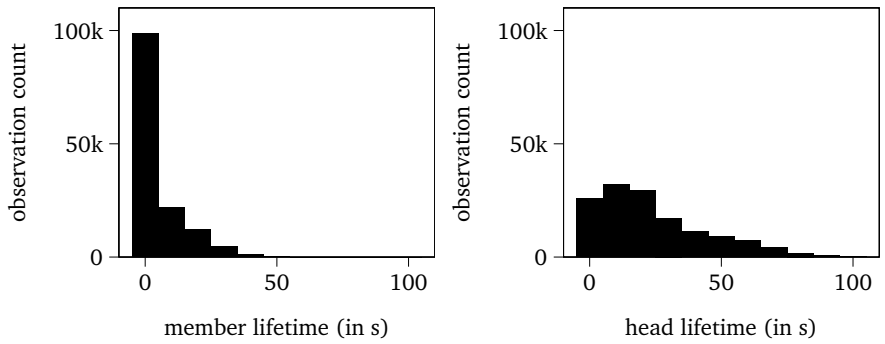


**Figure 2.6** – Exemplary results using the IEEE 802.11p stack of *Veins LTE*. Shown are how long cars are cluster members (left) and cluster heads (right); based on [49] © 2014 IEEE.

The performance of the clustering can be seen in Figure 2.6. Here, we investigate the clusters created using IEEE 802.11p. A car is only a Cluster Member (CM) for a short period indicating many small clusters with a short lifetime. Nevertheless, when we look at the time a car acts as CH we can see that clusters are able to exist for quite a long time, in some cases even longer than 50 s. This contradicts the impression from the CM graph as it indicates that clusters live for a longer time, but CMs change more frequently. Therefore, it seems that the underlying algorithm can maintain clusters for a longer time, as a result improving cluster stability.

Overall, both communication stacks still worked fine and the results provide basic insights into the clustering process. The long delays in LTE can be attributed to *SimuLTE* and its non-line-of-sight model, which makes receiving messages from farther away quite challenging. This leads to long delays, as the eNB regularly tries to send data again until it is successfully received. As the uploaded data gets larger with longer intervals, the chances of a message getting lost also increases leading to the observed large uplink delays.

# Chapter 3

# Car4ICT: Service Provision with Vehicular Networks

In the first two chapters, we presented smart cities and their characteristics. We focused on two of them: Information and Communication Technologies (ICTs) and mobility in the form of connected cars. Besides providing transportation, these cars offer additional exploitable resources: processing power, storage space and communication capabilities. As discussed in Chapter 1, smart cities use ICTs to generate, collect, and process of huge amounts of (sensor) data [2]. There is concern that current infrastructure will not be able to cope with these tasks, best recognized in the development of 5G [37]–[39]. Connected cars are another option to aid in performing these tasks by providing *virtual infrastructure*. They are not only capable of communication, but also provide processing and storage resources. Furthermore, they are not only able to simply handle the data, but also able to provide additional services, e.g., storage offloading, routing assistance, or internet access.

In this chapter, we focus on service provision for drivers, non-drivers, and machines. We present an architecture named *Car4ICT*, which exploits connected cars for service offering, discovery, and, finally, utilization in smart cities. When looking back at the outlined research questions (cf. Section 1.3), Car4ICT can be summarized as an architecture where connected cars enable and become an *Information Hub for Smart Cities*. The research presented in this chapter is based on the following peer-reviewed publications:

- O. Altintas, F. Dressler, F. Hagenauer, M. Matsumoto, M. Sepulcre, and C. Sommer, "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)*, Hong Kong, China: IEEE, Apr. 2015, pp. 654–659
  My contributions to this conference paper were the design, implementation, and evaluation of the proposed Car4ICT architecture. I was involved in developing the architecture based on the core premise: Cars being a service provider in future ICT systems. Based on potential use cases, I categorized the different Car4ICT entities and established a way to identify services using concepts from the area of Named Data Networking (NDN) and Information-Centric Networking (ICN). Finally, I implemented the architecture in our simulation environment and conducted evaluations with a focus on service discovery.

- F. Hagenauer, C. Sommer, R. Onishi, M. Wilhelm, F. Dressler, and O. Altintas, "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)*, San Francisco, CA: IEEE, Apr. 2016, pp. 788–793
  My contributions to this conference publication were the extension of the Car4ICT architecture for rural areas and freeways. I evaluated what concep-

tual changes are required in order to connect distant users with each other. As a result, the service discovery mechanisms and its underlying routing had to be changed. Furthermore, I focused on the parameterization of the Contention Based Forwarding (CBF) routing algorithm. For this, I created a realistic freeway scenario based on real world map and traffic data. This was then used to conduct a parameter study as well as an investigation into the effects of the penetration rate of equipped vehicles onto the performance.

- F. Hagenauer, F. Dressler, O. Altintas, and C. Sommer, "Cars as a Main ICT Resource of Smart Cities," in *Smart Cities and Homes - Key Enabling Technologies*, M. S. Obaidat and P. Nicopolitidis, Eds., Elsevier, May 2016, pp. 131–147
  My contributions to this book chapter were the extension of the Car4ICT architecture with additional features. These updates included better service comparison and improved differentiation of the various entities. Finally, I evaluated these extensions by adapting the existing implementation of the Car4ICT modules.

## 3.1 Motivation

As already discussed in Chapter 2, smart cities are a complex system with numerous goals, examples being happier citizens, prosperous business, or a more sustainable environment. One key building block for these cities are ICTs, which produce, collect, and process large amounts of data. Such generated data can subsequently be used as a source for a huge number of services, including environmental monitoring, smart grids, or smart mobility. Furthermore, such technologies are envisioned to enable faster and more resilient responses in disaster situations [2], [107].

As infrastructure struggles to keep up with demands like wireless capacity or data rate [37]–[39], these issues will only become more serious when more smart systems go online. Currently, a lot of resources are put into research and standardization of technologies, which should be able to cope with these huge amounts of data [37].

In this chapter we outline a potential solution to some of these issues by using connected cars as a main ICT resource. Beside transportation, we aim to exploit their sensors, networking capabilities, and processing power to support the foundations of a smart city. Therefore, connected cars are both a solution to the demand issue and an opportunity for new services:

- **Virtual Infrastructure**: While cars are moving, they are able to act like mobile base stations and enable users to connect to them. There are many potential users who can exploit this, e.g., people waiting for the bus, devices in smart buildings, or other connected cars. Even stationary sensors can connect and provide their data by uploading it to passing cars.

- **Resources**: Connected cars are able to form a network on their own. Such a network combines processing and storage capacities of all participating cars resulting in a huge amount of available resources. Again, these resources can be offered as a service and used by potential users.

- **Services**: Beside providing network access and resources, connected cars are an opportunity for further services. Such services include personal weather forecasts, driving route guidance, or exploiting dash cams for monitoring.

### 3.1.1 Basic Car4ICT Architecture

One main advantage of connected cars is that they will be ubiquitous in urban areas. Take for example the United Kingdom, where the average annual mileage per motor vehicle stays roughly the same while the number of vehicles steadily increases since the 1950s [108]. Even with London charging vehicles for entering the city center, overall urban road traffic in Great Britain keeps steady [108]. If a city bans personal

vehicles, there still will be cars in future cities, e.g., taxis, buses, ride-sharing cars, and delivery vehicles.

Connected cars can provide services mostly independent of time and geographic context. This is true even in emergency situations (e.g., after a hurricane or an earthquake) where they provide access to infrastructure outside of the disaster area [109]. Furthermore, we do not limit our services to a specific application or a specific category. On the contrary, our goal was to support arbitrary kinds of services only excluding services requiring a low delay, e.g., safety applications. We term our envisioned architecture for exploiting connected cars in smart cities *Car4ICT*.

There have been various approaches to design architectures turning vehicles into large scale networks [75]. These architectures often rely on a high market penetration of connected cars. This becomes a problem in the introductory phase as the fraction of equipped cars will only grow slowly [75]. Therefore, we designed Car4ICT to not rely on a single type of communication technology. For example, if the penetration rate of cars with ad-hoc technology is too low, users should be able to switch to a different technology, e.g., cellular networks for discovering and using services. Moreover, we designed and evaluated Car4ICT using different traffic densities to understand the implied effects.

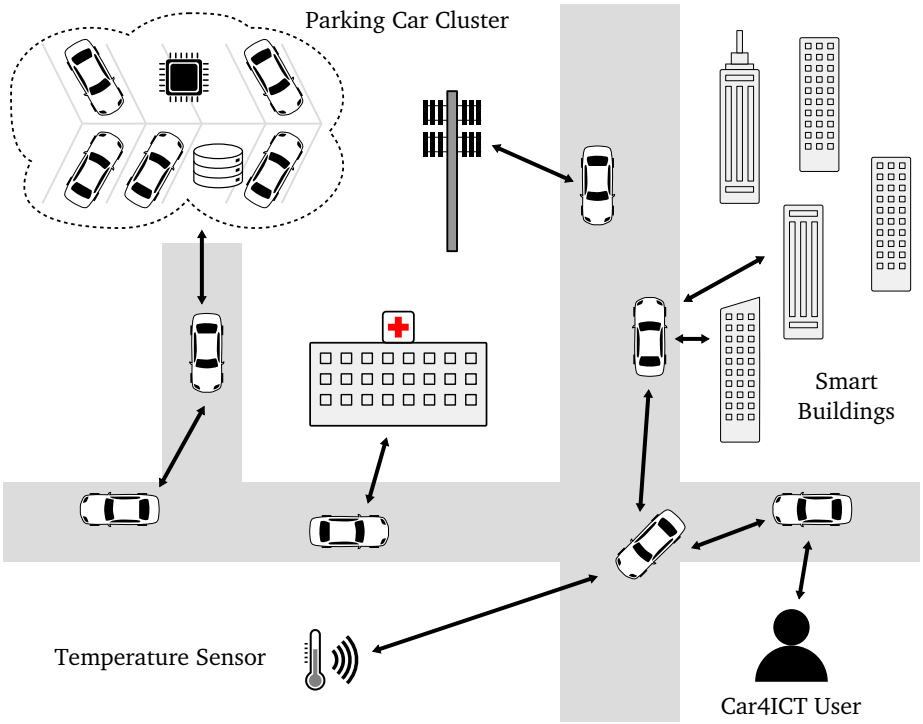


**Figure 3.1** – Overview of the Car4ICT architecture. It includes parked and driving cars and users offering and utilizing services; based on [50] © 2015 IEEE.

An overview of the envisioned architecture can be seen in Figure 3.1. On the left, we see a cluster of parked cars offering services like storage and processing power. On the bottom, there is a smart sensor offering temperature readings, which are promptly gathered by the user in the lower right corner. Furthermore, there is one car offering services via the cellular network and one collecting information from smart buildings. All of this is enabled by cars providing service discovery and provision throughout a smart city. The proposed architecture is flexible and extensible and consequently provides a base for many diverse applications (e.g., distributed processing, monitoring, or sensing).

### 3.1.2 Car4ICT in Disaster Scenarios

Another potential scenario for using Car4ICT are disasters. Many large cities worldwide are built in disaster-prone areas (e.g., Tokyo, San Francisco, or Houston). An earthquake or a hurricane has the potential to destroy a cities infrastructure and cutting of communication to rescue operations. In Figure 3.2 we outline how Car4ICT could help in such a situation. First, people affected by the disaster (right side) share messages destined for outside of the disaster area with a car. Moreover, connected cars collect data from the remaining functioning infrastructure, like autonomous smart sensors. This is done via ad-hoc Wi-Fi, LTE Direct, or any other potential technology not relying on existing infrastructure. Second, the car collecting the data ferries the data to the emergency coordination site. By simply driving, the car brings the message away from the disaster, essentially employing Store-Carry-Forward (SCF). Nevertheless, if other cars are available, the messages might be forwarded using ad-hoc communication technologies. After reaching the evacuation site, more

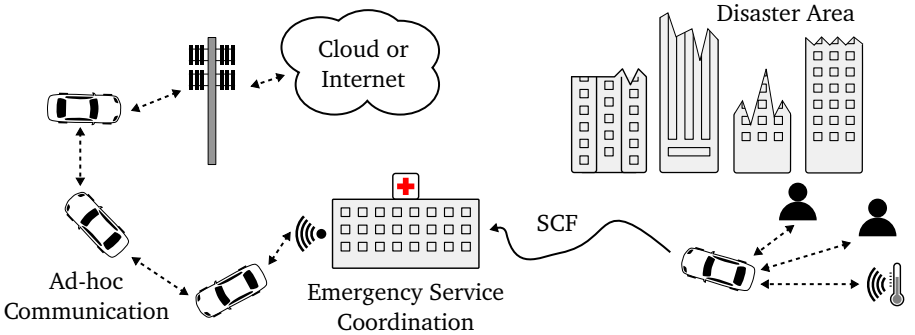


**Figure 3.2** – Concept of Car4ICT in an disaster situation with destroyed infrastructure. The right shows users connecting to a Car4ICT-enabled car which then drives with the data to an emergency service coordination site. Afterwards, using ad-hoc communication and cellular networks, the messages are delivered to a cloud or to the internet; based on [52] © Elsevier B.V.

cars are available and data is exchanged between them. Finally, a car establishes a connection to a cellular base station, i.e., working infrastructure. Now the messages are delivered to their destination outside of the disaster area via the internet or other cloud services. This all is possible because Car4ICT does not rely on infrastructure to work, but, if available, can exploit it to increase the performance.

The initial environment for Car4ICT were cities, i.e., urban environments. Due to their dense population, cities provide numerous potential services. Similarly, the number of users for these services is also large. Moreover, vehicles are abundant in cities, especially during daytime. All three properties combined — many services, numerous users, and cars with the necessary technology — make cities the perfect environment for Car4ICT.

### 3.1.3 Car4ICT in Rural Areas

Beside using Car4ICT in cities, we extended the concept to rural areas, i.e., the countryside and freeways. While cities already facilitate many services, an extension to rural environments provides additional opportunities. Users would be able to use services farther away, potentially in another city with active Car4ICT providers. Therefore, the number of available services would increase, which provides a user with more options to choose from.

Besides having more services, we envision novel services by incorporating long-distance links into Car4ICT. To illustrate this, take for example a long-distance journey along a freeway. Car4ICT supports a car in collecting traffic information for the journey ahead. Generally, this is not a time critical task, but allows the driver to avoid large traffic jams by warning him ahead of time. Similarly, a user would be able to create a personalized weather forecast for the route. This could be useful during winter with potentially slippery or snow-covered roads. To acquire necessary data, the user would utilize Car4ICT to collect weather information from sensors along the route. Like in urban scenarios, the necessary data could be provided by road-side sensors or by driving cars with the appropriate equipment.

The discussed new services do not necessarily need connected cars, but would mostly work using cellular networks. While metropolitan areas are covered well by cellular infrastructure, rural areas are not. In 2013, a report covering the eastern parts of Africa outlined that 80 % of the population are living in an area with cellular coverage [110]. Nevertheless, this mostly considers urban areas and therefore, around 50 % of the rural areas were not covered. The number was even lower for certain countries, e.g., down to 35 % in Kenya. While the European Union finds that 97.9 %/89.9 % of the urban/rural households are covered by LTE [40], this does not necessarily imply that reception in rural areas is useful. But, as a report in Germany found out, even available cellular coverage does not imply useful data

communication [41]. The authors were driving around the city of Frankfurt am Main, Germany and periodically tried to establish a connection with a server. They were not able to connect to the server 8 % of the time they tried to establish a connection. With an even worse infrastructure outside of urban areas, the failure rate can be expected to increase. Recently an interesting anecdote happened, as the German Federal Minister for Economic Affairs publicly called Germany's cellular network "embarrassing".[8] He even admitted he is not taking calls from foreign politicians while on the road. Therefore, we argue, that, by introducing Car4ICT in rural areas we can provide coverage without inducing large costs for building the necessary infrastructure. Car4ICT would complement existing cellular infrastructure to provide the users with a better experience.

Extending Car4ICT for rural environments and intercity connections also improves the resilience of communication links in disaster scenarios. It enables cars using Car4ICT to provide services for large disaster areas. While, due to SCF, the data transfer might take longer, cars are able to reach disaster-struck areas. Additionally, there is another independent communication link if a disaster disables infrastructure-based transmissions between two cities.

Several characteristics are different when comparing urban and rural environments, e.g., connections between users, or the mobility of cars. Due to the higher density of users in a city, a user offering a suitable service is usually closer than in rural areas. Therefore, different approaches for service discovery and communication need to be used between the Car4ICT entities. Moreover, cars follow different traffic patterns on free- and highways compared to urban areas [111]. This again requires another approach in the underlying Car4ICT architecture to successfully accommodate these use cases. Consequently, we provide an updated service discovery and data transfer approach based on Contention Based Forwarding (CBF) integrated into the Car4ICT architecture. With these modifications Car4ICT supports service discovery and provision in rural areas and along freeways. This provides users with numerous additional services. On the one hand, more variation of existing services and on the other hand completely new ones. To investigate the feasibility of these updates we simulated a realistic freeway in Japan. The scenario was built upon a real-world map and associated traffic data made available by Japanese authorities. We put the focus on delays for discovering services over various distances and different traffic conditions. This helps in understanding the performance and the limitations of Car4ICT in rural environments.

[8] `http://www.manager-magazin.de/video/peter-altmaier-bundesminister-schaemt-sich-fuer-handynetz-video-99023002.html`

### 3.1.4 Car4ICT Use Cases

Before outlining the concrete components of the architecture, we discuss several applications for Car4ICT. This should help to understand what kind of applications we envision to be used with Car4ICT. As discussed in Chapter 2, applications and services in vehicular networks can be categorized in different ways [19]–[21]. These categories almost always include a *safety* category with rather strict requirements, especially regarding delay. Although such applications would theoretically work with Car4ICT, we designed the architecture with *non-safety* applications as main use cases.

As Car4ICT supports different kinds of users, there are various use cases and services. One example are smart sensors detecting a natural disaster. In this scenario, there is the need to validate and process the data in order to plan and begin emergency measures. But, due to the disaster existing infrastructure might already be destroyed. In this scenario, Car4ICT can help in two different ways: (1) collect data from the sensors and transport them to a data center where it can be processed or (2) offload tasks by processing the data locally with the resources provided by the cars. To transport data to the outside of the disaster area, we rely both on the communication technology of cars and their mobility. Communication to forward messages to other cars until infrastructure is available again and mobility in case there is no other car in communication range. If this happens, the car can drive out of the disaster area until suitable forwarders or operational infrastructure is reached. By processing and storing data locally cars essentially offload resources from the internet or a data center. This becomes especially important in such a disaster scenario as the original infrastructure might not be reachable or destroyed.

A less dramatic example is a tourist who runs out of storage on its phone but wants to continue taking pictures. With Car4ICT, there is no need to rely on existing infrastructure — which might not be compatible — or use an expensive data plan. The tourist can connect to a Car4ICT-enabled car using widely available technologies like Wi-Fi or Bluetooth and request secure storage. Car4ICT takes care of discovering fitting storage and provides the tourist with a list of options. It chooses one of them and offloads her pictures. Later, when the tourist is back at the hotel, it can again use the Car4ICT network to get the offloaded pictures back. To do this, it requests them from any Car4ICT-enabled car passing by, which then takes care of discovering the pictures and returning them.

As an example for a smart automated system using Car4ICT, we outline a weather forecast system relying on local temperature and atmospheric pressure information. The system can use Car4ICT to search for appropriate sensors in certain regions. Cars passing through the region can be equipped with such sensors or fixed ones are installed. After Car4ICT provides the forecast system with a list of available

sensors, it can choose which data to acquire. Again, the data exchange is mediated by Car4ICT and enables the forecast system to gather large amounts of data in order to generate a detailed temperature forecast.

Recently, dashboard cameras are becoming increasingly common in new cars. Car4ICT enables to exploit these pictures for semi-live picture sharing. With a dashboard camera, a car can record videos and take pictures of its surroundings. Depending on the car's storage, it can keep such data and offer them via Car4ICT resulting in semi-live views of the area. Now, other users can search for this kind of pictures from a certain region and use Car4ICT to acquire the data in order to see how it recently looked.

When considering freeway scenarios, additional use-cases come up. For example, the car can provide information for the journey ahead to the driver. If the user enters its destination, the Car4ICT-enabled car acts as a consumer and queries providers along the route. Such providers, e.g., installed sensors or cars driving there, supply the car with various kinds of information such as traffic density, current weather situation, or road conditions. Based on this, the car recommends a route to the destination and modify the route while driving due to more available information.

### 3.1.5 Contributions

To summarize, this chapter covers the Car4ICT architecture both for urban and rural environments. It outlines the involved entities and how they are able to access the Car4ICT network. Afterwards, the algorithms for offering services and discovering them are discussed. The evaluation of the architecture focuses on service discovery, how to improve it in urban scenarios, and the effect of the penetration rate of equipped cars. Furthermore, we discuss the modifications necessary to use Car4ICT in rural environments and on freeways. Here, the emphasis is on the adapted routing scheme based on Contention Based Forwarding and Store-Carry-Forward and their respective parametrization. This part is evaluated using a realistic freeway scenario and by conducting a parameter study. Again, the penetration rate of equipped cars is investigated. The core contributions of this chapter are:

- First, we outline the architecture of Car4ICT, including involved entities, access procedures, and service identification (Section 3.3).
- Next, we evaluate service discovery in urban environments. Here, the focus lies on the effect of broadcast intervals and penetration rate of equipped vehicles onto the service discovery delay (Section 3.4).
- Finally, we discuss changes necessary to use Car4ICT on freeways. We do this by updating the service discovery algorithm and evaluating parameterization and penetration rate in a realistic freeway scenario (Section 3.5).

## 3.2 Preliminaries

The Car4ICT architecture is built upon ideas from many areas of vehicular networking research, including, but not limited to, Named Data Networking, Information-Centric Networking, routing, and Service Discovery Protocols. In the following we outline related work in these areas and how we differentiate from or build upon them.

Using the additional resources of connected cars to form networks has been investigated, e.g., by Gerla [32] or Olariu, Hristov, and Yan [33] proposing the concept of Vehicular Clouds. Like Car4ICT, such concepts envision cars as gateways for multimedia streaming, urban sensing, providing internet access to outside users, or enable emergency services, most of the time focusing on a single use case. For example, Barros [112] discuss a service where internet access is provided using mesh-routing between connected cars. Or, Baron et al. [113], who built an architecture for delay tolerant data transfer on top of a vehicular network. Dressler, Handle, and Sommer [89] outline an architecture to exploit parked cars. These cars are organized using the Virtual Cord Protocol (VCP). Along this cord, they are interconnected with each other. Users store and retrieve data from this network of connected cars. Car4ICT takes such ideas one step further by being a large-scale vehicular network providing different kinds of services to users. Such users are not only cars, but also people connected via smartphones, or machines (e.g., processing sensor data collected from a smart city).

### 3.2.1 Service Discovery Protocols

A major part of the Car4ICT architecture can be categorized as a Service Discovery Protocol. Such algorithms enable the discovery and (and potentially the utilization) of services. Approaches specific for Mobile Ad Hoc Networks (MANETs) have been investigated over the years [114]. Still, there are few approaches mainly designed or adapted for vehicular networks. Sailhan and Issarny [115] propose one of the earliest examples based on a virtual overlay network. The goal was to reduce the load on the channel by dynamically adding and removing service directories. For describing services, the Web Service Description Language (WSDL) was used. Sharing and discovering services was done via four geographic trajectories. This works well in a Manhattan grid scenario, but might lead to issues if other street layouts are considered. If the city has a different layout, the junctions and corners could lead to service discoveries running into dead ends. The evaluation was done using randomly moving nodes with a speed of 1 m/s. Therefore, it is unclear how the SDP works in realistic Vehicular Ad Hoc Networks (VANETs) as cars usually move faster and in a more specific pattern. Specifically designed for VANETs is the SDP proposed by Abrougui, Boukerche, and Pazzi [116]. Their approach necessitates infrastructure

support in the form of Roadside Units (RSUs). Services can take one of three different forms in their architecture: moving services, fixed services, and migratory services. The last form are services, which generally act like fixed services, but if no resources are available, they can be moved to other locations by cars. Beside the three standard actors in a SDP, service provider, service consumer, and service directory, Lakas, Serhani, and Boulmalf [117] propose an additional actor, namely mobile service directory. Those are cars moving around while still acquiring service information and providing it to users. Noguchi et al. [118] present another SDP for vehicular networks. Based on IPv6 and geonetworking, their approach enables users to find services in a specific geographic area. To make their proposed system work, they rely on roadside infrastructure.

To make an SDP like Car4ICT work, there is the need to identify services and map them to locations. Research in the area of future internet faces a similar issue, namely content identification. More formally this refers to associating content with unique names, i.e., an identifier. One approach of the future internet research is Information-Centric Networking [119]. It has been combined with vehicular networks by Amadeo, Campolo, and Molinaro [120]. They build a custom protocol enabling ICN on top of IEEE 802.11p. By doing this, they do not rely on any form of IP. Another future internet approach is Named Data Networking [121], which has been proposed for vehicular networks among others by Wang et al. [122] and Grassi et al. [123]. Their approaches work only with a limited set of applications and the proposed systems have been optimized for them. Therefore, it is not clear how well the systems work if more applications use the same resources, especially regarding the wireless channel. Melazzi et al. [124] propose an architecture called *Internames*, which aims to be technology agnostic and works with different types of networks: IP-based, cellular, or ad-hoc networks. *Internames* is based on mapping service identifiers to locations and associating this information with the technologies needed to use the services. Nevertheless, *Internames* focuses on a limited set of applications. Another approach is provided by the IETF, who identify resources using an extended URI concept [125].

### 3.2.2 Rural and Urban Differences

Cars moving on a freeway have a different mobility pattern compared to cars in urban environments. Therefore, we need to adapt parts of the Car4ICT architecture in order to better support rural areas. Especially service discovery and its underlying routing needs to be adjusted with a focus on reaching cars/services farther away.

There are numerous routing algorithms available to be used in VANETs. But none of them sufficiently cover urban as well as freeway scenarios. This is backed by Fonseca and Vazão [126], who investigate multiple routing algorithms specifically

designed for VANETs on their suitability to be used in urban and freeway scenarios. They conclude that currently no satisfactory algorithm exists. According to the authors, this necessitates that either a new one must be developed to cover both aspects or to use two different ones and switch between them when necessary.

We base our adapted Car4ICT protocol on Contention Based Forwarding (CBF), first outlined by Füßler et al. [127]. Whenever a node receives a message while using CBF, it initially buffers the message before forwarding it. The closer the car is to the destination, the shorter it keeps the message in a buffer before sending it again. Other cars overhearing a forwarded packet see this as an acknowledgment and discard it. The underlying idea is to have the car closest to the destination forward the packet first without explicitly coordinating with any other recipients.

The initially proposed version of CBF worked well on freeways, but did not in urban environments due to not incorporating the road topology — which is more complicated in cities compared to freeways — into the decision process [128]. Therefore, various studies have been conducted trying to improve the performance. One example was done by Al-Kubati et al. [129], who identified crowded intersections as an issue. At these positions, many cars would simultaneously try to forward a message leading to packet collisions. To avoid this, the authors added a factor for deciding who forwards the packet: the number of potential directions the car can send into. While they claim that their improved CBF works in urban and freeway scenarios, the authors only evaluated it using a Manhattan grid. Another example of a CBF improvement was done by Salvo et al. [130]. Beside relying on distance, they added an angle parameter to the forwarding decision. Based on this, the best suited car is chosen. Their evaluation was also done in a Manhattan grid scenario and not on a freeway.

Beside these attempts to improve CBF, the European ETSI ITS-G5 standard also includes a version of CBF [131]. This version does not try to fix issues in urban environments, but rather adds more parameters increasing the flexibility of CBF. In the original CBF algorithm [127] packets were only assumed to travel a certain maximum distance. If a car received a packet from farther away, it discarded the packet. This has been omitted in the version provided by the European Telecommunications Standards Institute (ETSI). Furthermore, they added a minimal buffer controlling how long a car keeps a packet before sending it onward.

## 3.3 The Car4ICT Concept

Car4ICT makes only few assumptions about the vehicles it relies on. It basically assumes that they are equipped with communication technology to connect to each other. Additionally, to be able to assist in service discovery, cars should be equipped with processing and storing capabilities. Any kind of existing infrastructure is not necessary for Car4ICT to work. Although, if infrastructure like cellular networks is available, it can be exploited to improve the performance. Furthermore, Car4ICT should not replace stationary networks, but complement available networks by being able to use Wi-Fi Access Points (APs) or cellular networks if available.

Which technology entities use to communicate with each other is not mandated by Car4ICT, but is subject to availability. Depending on certain aspects like transmission delay, connection stability, or incurring fees, a node can choose the most suitable technologies. If the user is a person, it can make a conscious choice, which is not the case for a machine. Therefore, there needs to be a mechanism to automatically choose the best suitable technology for Machine-to-Machine (M2M) communication. There might be even different communication technologies in place between the entities. For example, we recommend cars to use IEEE 802.11p when communicating with each other, the current base for vehicular networking standards around the world. But, this might not be the best suited technology for users, which more likely have access to Wi-Fi or Bluetooth instead of IEEE 802.11p. Therefore, Car4ICT-enabled cars could accept connections from users via Wi-Fi, but internally use IEEE 802.11p, essentially making Car4ICT a heterogeneous network. Another option, especially relevant for in-car subsystems, is to be connected via wire. This enables in-car sensors to provide data via Car4ICT or programs to acquire information (e.g., traffic information or weather sensor readings) to improve the driving experience without the need to rely on wireless communication.

We envision connected cars as the main building block of ICT systems in smart cities. Our solution revolves around services, especially how to find and use them. Potential users of the system include smart devices and people. Service examples include offering storage space and processing power, or access to smart sensors (either in the car or external). The workflow of using Car4ICT is as follows:

- **Access**: Before using the Car4ICT network, users need to access it. Two things need to happen to achieve this: (1) users need to be able to connect to the network and (2) they need to be verified.
- **Offer**: After accessing Car4ICT, if a user intends to offer a service, it needs to send the required information to a connected car. To make both users and cars understand service definitions, it is necessary to identify services using a common nomenclature.

- **Discover**: Similarly, a user might want to use a service. The first step is then to find suitable services and select a fitting one.
- **Utilize**: After discovering services, the user needs to actually use the services. Car4ICT provides the connection between users.

Connected cars take the role of a coordinator by storing available services in service tables and assist users in the discovery and utilization process. This is done by matching requests to available services. Car4ICT supports two approaches how service tables are stored: (1) they are not shared with other cars, but rather requests are sent to neighbors to discover potential matches, or (2) service tables are actively shared between cars. Which way to use depends mostly on the environment, i.e., if it is urban or rural. While it might make sense to share services between cars in cities, this does not scale well on freeways. Independent of the approach, whenever a match to a request is found, a list of results is compiled and sent to the requesting user. The user is now able to select a service provider and exchange data with it — again coordinated by Car4ICT. Such an exchange can be done in multiple ways, either by routing or by exploiting the mobility and using SCF.

Regarding terminology in Car4ICT, we describe **Users** as entities making use of the Car4ICT network. That can either be **Providers** offering a service or **Consumers** discovering and utilizing a service. In doing that, users are assisted by **Members**, which handle the flow of control and data messages throughout the architecture. To describe services, we rely on **Identifiers**. They can both be used to describe a service and as a query when searching for available services.

### 3.3.1 Car4ICT Entities

The whole Car4ICT architecture is formed using three types of entities: *consumers*, *providers*, and *members*. Consumers and providers are also categorized as *users*, as they utilize the architecture provided by the members.

**Providers** are users providing a certain service. They connect via a member to the Car4ICT network and supply the network with information about their offered services. A provider can be a person using a device to connect to the network (e.g., a smartphone or a notebook), a machine (e.g., a smart sensor offering data), or a subsystem inside the car. Providers announce offered services by sending *Offer* messages to members.

**Consumers** are users utilizing one of the services available in the Car4ICT network. Like providers, they connect via a member. Usually, they do not directly make use of a service, but need to discover available services beforehand. Afterwards they rely on Car4ICT to be connected to the provider in order to use the service. Again, similar to a provider, a consumer can either be a person, a machine, or a

subsystem of a Car4ICT-enabled car. Consumers start a service discovery process by sending *Request* messages to members and after choosing a suitable provider communicate with them using *Data* messages.

**Members** are the core of the Car4ICT architecture. They are connected cars essentially forming the backbone of the Car4ICT network. Their main task is to handle the control and data message flow throughout the network. Furthermore, they are also in charge of providing the necessary means to access the Car4ICT network. Typically, a member is a driving or a parked connected car. Note that, it is also possible to deploy additional infrastructure nodes taking the role of a member. Their functionality covers multiple tasks, some done periodically and some triggered by receiving a certain message:

- If the member receives an *Offer* message from a provider, the identifier of the offered service is stored in its local service table. Usually, (stationary) providers need to regularly update their offers to avoid expiration. Therefore, if a member receives a new service, it compares it to all stored services. If there is already a matching service, the existing entry is replaced with it, otherwise a new entry is added.

- If the member receives a *Request* message from a consumer, the included identifier is treated as a query. By applying the query to the service table, the member determines matching services. If none are available, the member might rely on additional approaches to extend the search (e.g., forwarding the *Request* to neighbors, or query a central service table).

- If a member receives a *Data* message from either a consumer or a provider, it tries to forward the message. Depending on the availability of a route towards the destination, the member sends the message or stores it to deliver it later. Note that, forwarding in rural environments is different to urban environments, which is discussed separately in Section 3.5.

- Periodically, Car4ICT members purge outdated entries from their service tables. This is based on two pieces of service information included in every identifier: geographic position and validity. As the validity describes the expiration time of a service, all expired entries are deleted. Similarly, if a service is too far away, it is removed as well. Strategies for determining the distance to a service include counting the hops to reach it or the geographic distance to it.

- If activated, a member regularly sends a broadcast including its service table. This can be done in a separate message or by piggybacking it to an existing regular control message. Such a broadcast is used to inform other members of the current available services and allows them to integrate this information into their own service tables.

### 3.3.2 Access Procedures

To prevent abuse of the system, we assume that all users have been verified before being granted general access to the Car4ICT network. Only after acquiring access, they can offer services or discover and utilize them. We do not further specify what is needed to be verified as this is out of the scope of this thesis. As an example, Public Key Infrastructure (PKI) could be used. Each user gets a key after being verified by the responsible authority. If a consumer or provider wants to connect to the network, they can present the key to the member and be granted access. Furthermore, a member can issue a temporary grant, which enables faster access on subsequent connections and removes the overhead of verifying the key.

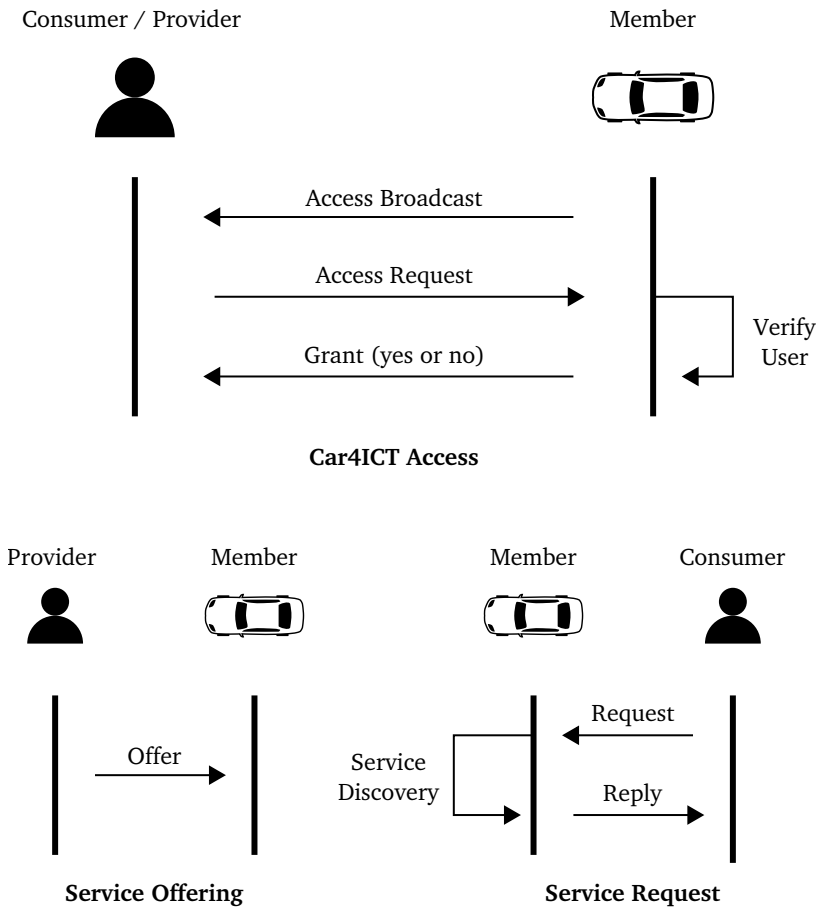


**Figure 3.3** – Overview of users utilizing Car4ICT: accessing the network, offering a service, and searching for and using a service; based on [50] © 2015 IEEE.

In Figure 3.3 we show an overview of the access procedures between members and users. For this, there is no difference between providers and consumers. Initially, members announce their availability by using periodic beacons, i.e., *Access Broadcasts*. Users receive these beacons and use them to detect members in the surroundings. This assumes that users and members communicate using the same technology. After finding a suitable member, the user sends an *Access Request* containing their issued key and a unique identifier. On receiving such an *Access Request*, the member verifies the key (e.g., by using PKI). If the verification was successful, the user is granted access to the Car4ICT network. Otherwise, the user is rejected and is not able to use Car4ICT. In any case the user receives a *Grant* message containing the result of the verification. If access is granted, the user might also receive a temporary grant, which can be used to bypass the initial verification. This grant has a Time to Live (TTL) to avoid misuse and is accepted by any Car4ICT member. On special occasions, such grants can be issued with an infinite TTL. This is useful if the user is connected via wire to the Car4ICT member, e.g., a subsystem in a Car4ICT-enabled car. Afterwards, the user is able to send *Offer* messages to any member to advertise available services (lower left). Furthermore, if the user wants to exploit a service, it must send a *Request* message to any available member (lower right). Based on the *Request* message content, the member starts the service discovery process and sends a reply containing available services.

### 3.3.3 Identifying Services

As already outlined, one main component of Car4ICT are the service tables. They include information about available services and who offers them. To make service tables useful, they need to be regularly updated with new information. Users who offer a service are therefore responsible to periodically announce them. We assume that such announcements can be done efficiently by piggybacking them onto already existing periodic messages. Such messages are part of the different standards, e.g., Cooperative Awareness Messages (CAMs) as part of ETSI ITS G5 in Europe [68], or Basic Safety Messages (BSM) as part of IEEE WAVE [81] in the U.S. Generally, these messages are used to exchange neighbor information, but there is no reason to not extend them with support for other applications. Beside service announcements, service tables themselves can be attached to such messages. To keep services longer available, members need to regularly hear from the provider. Otherwise, they would remove it from their service table. Therefore, exchanging service tables helps in keeping services available. The same is valid for services relevant to a specific geographic area — if a member leaves a certain area, it purges all invalid services, as no good performance is to be expected.

For efficient identification of services, we rely on a concept called *identifiers*. They are used to identify services, locate services, determine their validity, and add metadata to them. Still, their basic structure is rather simple: a single hash tag and a set of metadata. The hash tag is a string identifying the service. Examples are a hash of a file, which is offered by a provider or a well-known reference string for a resource. Such a reference could be for example `STORAGE`, referring to storage offloading or `CPU` if processing resources are offered. Metadata annotates the service with additional information.

There are two mandatory metadata fields included in every identifier: the service providers geographic location and the validity of the service. The geographic location, e.g., provided by an Global Navigation Satellite System (GNSS) like GPS, is used to limit the service to a geographic area. Moreover, consumers can use this to restrict their search to services from a specific area. Similarly, providers can request for their service to be only available in a certain region. Furthermore, if a consumer receives a list of discovered services, it can choose the closest service. Finally, the location can be used to purge services if the member travels too far away from the service consumer. In this scenario, the distance would add additional delay and therefore potentially degrade the service performance. The validity describes how long the service offer is valid, i.e., its TTL. This is done to keep service tables clean as it would be difficult to otherwise determine broken services.

Note that, service tables are not limited to members, although mostly they will be stored there. One could think of a central service table reachable via cellular networks. This would enable services from all around the world to be discoverable.

In Table 3.1 we show an example of a service table with identifiers of different types. The image offered by user 1 (i.e., file0) is only offered in Innsbruck and is valid for 5 more minutes. Two users offer a text file (file1), but each one is at a different location. The offer by user 12 is only valid for 10 minutes, while the one in Paderborn can be acquired for 1 more hour. A video is offered by two users, one located in Innsbruck and one in New York. The user in New York provides more metadata as the size of the video (2 MB) is included. Besides offering files, Car4ICT

**Table 3.1** – Car4ICT service table with various hash tags. Note that, while *location* and *validity* are mandatory, others, e.g., *type*, are not.

| User | Hash tag | Metadata | | |
|---|---|---|---|---|
| 1 | hash(file0) | location = Innsbruck, validity = 5 min, | type = image, | size = 4.0 MB |
| 12 | hash(file1) | location = Innsbruck, validity = 10 min, | type = text file, | size = 0.1 MB |
| 8 | hash(file1) | location = Paderborn, validity = 1 h, | type = text file, | size = 0.1 MB |
| 1 | hash(file2) | location = Innsbruck, validity = 1 d, | type = video | |
| 42 | hash(file2) | location = New York, validity = 1 d, | type = video, | size = 2.0 MB |
| 19 | `CPU` | location = Paderborn, validity = 12 h, | type = AMD | |
| 19 | `Storage` | location = Paderborn, validity = 12 h, | type = hours, | size = 9.0 GB |

also enables services intended to offload resources. In the example, user 19 offers both processing power (hash tag `CPU`) and storage (hash tag `Storage`). Beside the mandatory metadata, the user also includes the CPU architecture, the amount of available storage and how long data can be stored.

Identifiers in Car4ICT can be used in two ways, as service offers and as a query to request a service. The first kind is used to describe an offered service and is used in *Offer* messages and when storing identifiers in service tables. When discovering a service, the second kind is used as it additionally allows omitting mandatory metadata. Matching a query with service table entries necessitates comparison of these two kinds of identifiers. As already seen in Table 3.1, identifiers sent by different users might describe the same service while having individual metadata.

Formally, we denote an identifier $I_x$ as a tuple $(h_x, \mathbb{M}_x)$, where $h_x$ is the hash of the identifier and $\mathbb{M}_x$ the set of all metadata. An entry in the metadata table is described as $(k, v)$ with $k$ and $v$ being the respective key and value. Furthermore, the set of all identifiers is denoted by $\mathbb{S}$ and usually referred to as the service table. If a query is received by a member, the member matches it against $\mathbb{S}$ and returns the result set $\mathbb{R}$. We propose to use different methods to compare two identifiers, all of which are needed when using Car4ICT:

- **Hash only ($\overset{h}{=}$)**: To apply this comparison, the metadata can be omitted. This is useful if the user searches for a service without any special requirements, e.g., looking to offload processing tasks without requiring a specific processor architecture. Formally, the comparison is $I_1 \overset{h}{=} I_2 \leftrightarrow h_1 = h_2$. When applying a query $I_q$ to the service table $\mathbb{S}$ we get the result set

$$\mathbb{R} = \{I_q \overset{h}{=} I_x \quad | \forall I_x \in \mathbb{S}\}. \tag{3.1}$$

- **Subset matching ($\subseteq$)**: Using this comparison, metadata subsets are matched. Therefore, we require the hash to be equal and one subset of metadata included in the other identifiers metadata, but not vice versa. This is used when a user needs certain requirements to be fulfilled, but does not care about others. For example, if a user wants to use processing power with a certain number of cores, the query could look like this:

$$(h : \text{CPU}, \mathbb{M} : \{(k : \text{cores}, v : 4)\}). \tag{3.2}$$

  To put this more formally, $I_1 \subseteq I_2 \leftrightarrow (I_1 \overset{h}{=} I_2) \wedge (\forall m \in \mathbb{M}_1 : m \in \mathbb{M}_2)$. Similarly to $\overset{h}{=}$ we get the result set $\mathbb{R}$ when applying the query $I_q$ to the service table $\mathbb{S}$:

$$\mathbb{R} = \{I_q \subseteq I_x \quad | \forall I_x \in \mathbb{S}\}. \tag{3.3}$$

Note that, this query also supports a special mode where the hash tag is omitted. This allows to just search for specific metadata entries, for example file types when downloading data. Generally, this option should be used carefully as it has the potential to return a large result set $\mathbb{R}$. If the query omits the hash tag, then the comparison would be $I_1 \subseteq I_2 \leftrightarrow (\forall m \in \mathbb{M}_1 : m \in \mathbb{M}_2)$.

- **Equals (=)**: This comparison is the strictest one as it requires the hash and all metadata to be identical. Car4ICT relies on this method when retrieving a previously offloaded file as it makes sure that the correct data is re-acquired. Formally, this can be written as $I_1 = I_2 \leftrightarrow (I_1 \subseteq I_2) \wedge (I_2 \subseteq I_1)$. Again, the result set $\mathbb{R}$ is acquired by applying the query $I_q$ to the service table $\mathbb{S}$:

$$\mathbb{R} = \{I_q = I_x \quad | \forall I_x \in \mathbb{S}\}. \tag{3.4}$$

- **Geographic comparison ($\overset{h_r}{=}$)**: Sometimes a user wants to limit the search to a geographic area. To achieve this, the user can add a radius $r$ as a metadata entry to the query. If a member now searches for a service, entries outside the radius are discarded. As all comparison methods build upon $\overset{h}{=}$ it is only necessary to replace $\overset{h}{=}$ with $\overset{h_r}{=}$: $I_u \overset{h_r}{=} I_x \leftrightarrow (h_u = h_x) \wedge (\texttt{dist}(\text{position}_u, \text{position}_x) <= r)$, where `dist` calculates the distance between two geographic positions.

### 3.3.4 Providing and Consuming Services

Before using services provided via Car4ICT, a user must acquire a grant. This is depicted in Figure 3.3, with the lower part showing offering and discovering services. After getting the grant, there is a distinction between providers and consumers. A provider sends *Offer* messages including identifiers of the provided services and the issued grant. The member proceeds by storing these service in its own service table.

Similarly, a consumer sends a *Request* message also including a service identifier and the grant. But, in this case, the identifier is seen as a query used to discover available services. It triggers a search for matching services either in the local service table of the member or in the Car4ICT network, depending on the methodology.

The flow of control messages in the Car4ICT network can be seen in Figure 3.4. It shows consumers, providers, and members exchanging control messages. One example is a consumer requesting access to the Car4ICT network from a member and getting it granted (upper right). On receiving a successful grant, the consumer immediately follows up with a service *Request* message. The receiving member checks its local service table for relevant services and replies with a list of the found services. Afterwards, the dashed line indicates the flow of the data messages between provider and consumer. A provider is shown to offer sensor information, in this case weather data. It already received a temporary grant and therefore, it is not necessary

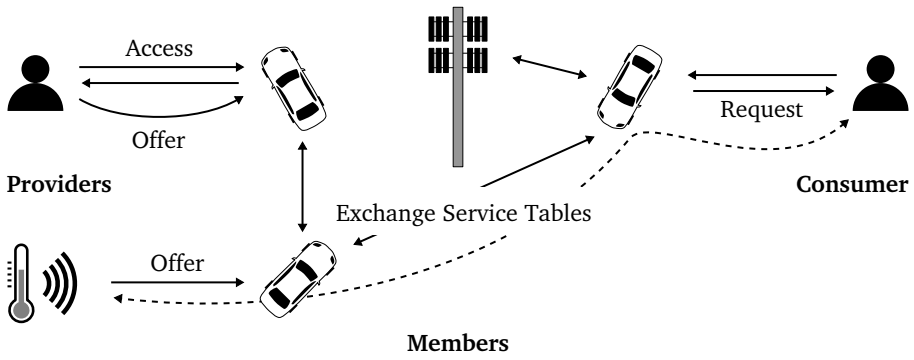


**Figure 3.4** – Examples of the control messages exchanged by Car4ICT. Shown are both kinds of users, providers (left) and a consumer (right), and members (center); based on [50] © 2015 IEEE.

to perform the access procedure before sending its offer. Furthermore, the Car4ICT members proactively exchange service tables to provide fast responses to service requests. Other shown examples include a provider offering smart sensor data.

#### 3.3.4.1 Service Discovery Approaches

Our proposed Car4ICT architecture supports a wide range of possible services, e.g., offloading processing resources, data storage, sharing sensor readings, or exchanging messages with a specific geographic area. Whenever a user requests a service, the member initially checks its own service table if a fitting service exists. If at least one matching service is found, a response is sent back to the user containing the identifiers of all found services. As might be expected, not all cars are aware of all services in the network. This leads to members not being able to successfully answer requests while other members would be. If this happens, there are multiple options to improve the response:

- To avoid larger delays induced by additional queries, members can proactively exchange their service tables. This would be done by periodically sending a broadcast including its own service table to neighbors. On receiving such a message, the member updates its own table with new or updated services. This reduces the response time of a request but can lead to significantly increased channel load, especially if there are many services available. Regarding urban environments, this is the preferred approach if the load on the channel is low enough. Otherwise, adaptive measures [27] could be used to reduce the load.

- If a member does not find a suitable entry in its service table it can forward the request to its neighbors. This can easily be done by sending a broadcast message. Receiving members are able to check their own service table for

matching entries. If a service is found, a reply is sent back to the originating member. Otherwise, there is the possibility to forward the request even further. Any successful replies received by the originating member are then forwarded to the consumer. Afterwards, the algorithm proceeds as usual with the consumer choosing an appropriate service and initiating the connection to the provider. This approach needs to be carefully implemented to avoid a broadcast storm. Generally, this method reduces the load on the wireless channel compared to exchanging complete service tables. But, it potentially increases the delay for the reply (e.g., if many Car4ICT members need to be contacted before a suitable service is found). This approach is useful in rural and freeway environments, where sharing service tables is usually not feasible due to the large amounts of data this would create.

- If the Car4ICT network has access to deployed infrastructure, like a cellular network, it could use a central server as an additional combined service table. Based on incentives (e.g., user payment, or longer-lasting services) a member uploads service information to this central service table. If a consumer sends a request, the member would forward it to the central server. There, services are matched and the result is returned to the member who forwards it to the user. Note that, using such a central service table, increases the importance of location information when choosing the appropriate service provider.
- It is also possible to combine the approaches mentioned above. This can be done either in parallel or sequential. For example, a member could in parallel forward a request to its 1-hop neighbors and query the central server. If done sequentially, a member could first check the local table, and, if this was not successful, forward the query to its neighbors. If this still proves to be unsuccessful, the final straw could be a query via LTE to the central server. This might even be coupled with a monetary incentive by the consumer.

If the search for a fitting service was successful, the requesting user receives the list of users offering the service. This list is accompanied by the service identifiers including metadata. Based on this list, the user can decide if and which service to make use of. When comparing the different user types, i.e., persons and machines, there are different reasons to use a specific service provider depending on what they provide. For example, take *storage-as-a-service*. On the one hand, a machine might opt to use a provider offering a long uptime. On the other hand, a person looking to store data aims to transfer the data fast and therefore chooses a provider in proximity. But, no matter what service the user chooses, the data exchange between user and provider is done by the members of the Car4ICT network. As every service table entry includes a source, the path from provider to consumer can be traced back and used for service fulfillment.

#### 3.3.4.2 Neighbor Tables and Data Transfer

Beside the service table, Car4ICT members also build a neighbor table. Information is added or updated whenever a member receives a message (*Offer*, *Request*, *Data*, or any kind of periodic broadcast). Each entry includes at least the following information of the sender: the geographic position, an ID uniquely identifying the sender, and the current time. If an entry associated with the sender already exists, position and time are updated.

The neighborhood information is stored to be used in routing *Data* messages and replies to forwarded *Request* messages. Generally, Car4ICT does not dictate a specific routing protocol, as this depends on the used technology and potentially available infrastructure. For example, cellular networks might rely on a different strategy compared to VANETs. And even there, another routing algorithm might be used if RSUs are available.

But, as we already have a neighbor table, it is possible to use a form of rudimentary routing. Based on the unique ID, every service table entry can be matched to a neighbor table entry. Using this information, any message can be sent towards the destination if the neighbor table includes the necessary information. Furthermore, it does not matter which kind of message was the base for the neighbor table entry.

In Figure 3.5 we show an example how this works in practice. First, the provider $p$ would announce his services via an *Offer* message to member $m_0$. On receiving the *Offer*, $m_0$ fill its service and neighbor table. Periodically, $m_0$ share its service table and that is how member $m_1$ acquires the service information. If consumer $c$ requests this information, it learns that $p$ offers the service. Therefore, if $c$ intends to use service $i_0$, it needs to send a *Data* message with destination $p$ to $m_1$. Based on the service table entry, $m_1$ deducts that $m_0$ is the next hop for provider $p$. Similarly, $m_0$

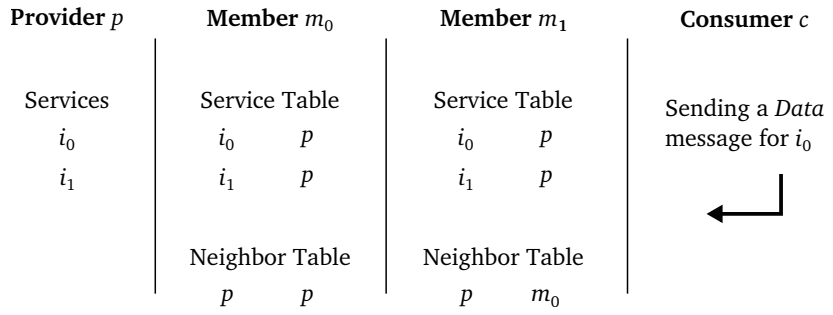


**Figure 3.5** – An example of the basic Car4ICT routing where a consumer $c$ sends a *Data* message for service $i_0$. Note that, two $p$ entries indicate a 1-hop connection between member $m_0$ and provider $p$.

forwards the *Data* message to $p$. Note that, when the *Data* message travels through the network $m_0$ and $m_1$ fill their neighbor table with information about $c$. These entries are later also used to route messages from $p$ towards $c$.

## 3.4 Evaluation of Car4ICT in Urban Environments

Car4ICT was initially designed with urban environments in mind. The abundance of potential services, users, and cars makes smart cities the perfect setting. Different to rural environments, there are no adaptations necessary to operate Car4ICT in cities.

For the simulations we used Veins LTE, mostly relying on the IEEE 802.11p part. Nevertheless, we added a *Connector* module located between the members *Car4ICT* module and the Veins LTE *Decision Maker*. It is an extension of the decision maker as it abstracts the technology even further. It supports wired connections as well, which are used to include in-car modules. Such devices provide, for example, sensor data or acquire route information using Car4ICT.

Our evaluation focus was the service discovery performance as it is the core of Car4ICT on which all other functionality depends. We assess the delay between the first sent *Request* and its response in increasingly worse conditions. This is done by reducing the number of cars being a Car4ICT member. Therefore, the probability of control messages reaching other members is reduced. In the evaluation, members announce their presence to the surroundings and proactively share their service tables. Users are placed in random positions around the Manhattan grid. Several providers offer data and a few consumers request the data. To measure the time until a service response is received, a consumer periodically requests a service until he acquires a successful reply. We measure the time between the first attempt and the fulfillment of the service request as *discovery latency*.

We implemented all outlined identifier comparison methods. The simulation mostly relies on $\overset{h}{=}$ and $\subseteq$. Comparison of hash tags ($\overset{h}{=}$) can be used by the user to get an overview of offered services. Subset matching ($\subseteq$) is then used to request the correct service. The complete comparison $=$ is only used when a member receives a new offer. If this happens, $=$ is used to update potential similar entries.

Our scenario of choice is a Manhattan grid with different densities of equipped vehicles, i.e. 35–415 per km$^2$. The dimensions of the building blocks are not simple squares but reflect real-world dimensions of downtown Manhattan (see Figure 3.6). These buildings also prevent radio transmissions, therefore restricting most of the communication to line-of-sight along the streets. Our scenario consists of 15 Car4ICT users placed at random positions throughout the grid. Ten of them are providers offering two kinds of services. Five offer a service requested by five consumers. The other five offer a different service to provide background noise. As a service we use

**Table 3.2** – Used simulation parameters.

| Parameter | Value |
|---|---|
| Simulated area | 0.7 km$^2$ |
| Average number of equipped cars per km$^2$ | 35 . . . 415 |
| Total number of users | 15 |
| Number of users requesting | 5 |
| Number of users offering | 5 |
| Inter-Vehicle Communication technology | IEEE 802.11p |
| Inter-Vehicle Communication maximal transmit power | 10 mW |
| Simulation duration | 80 s |
| Service table broadcast interval | 0.1 s . . . 10 s |
| Neighbor table entry lifetime | 10 s |
| Service table entry lifetime | 10 s |
| User request interval | 2 s |
| Request timeout | 30 s |

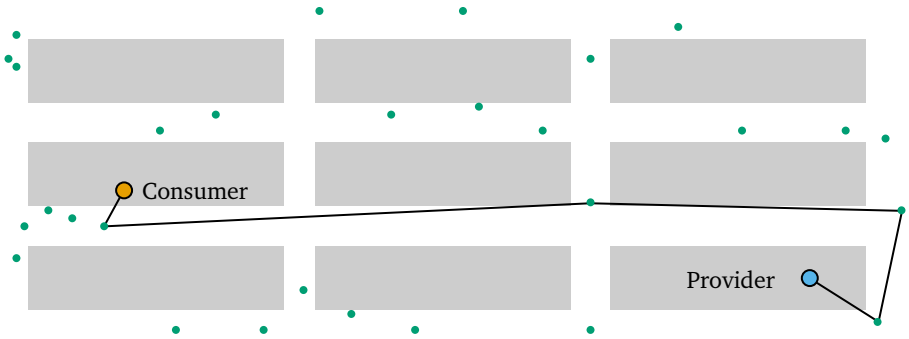


**Figure 3.6** – An excerpt of the used Manhattan grid scenario. It also includes a successful connection between a consumer and a provider; based on [52]


secure storage, i.e., a consumer requests previously stored data from the provider. Further parameters used during the simulation can be found in Table 3.2.

We show the results in the form of an eCDF, making it easier to see the performance of the various configurations. Furthermore, to measure the delay we do not only consider successful replies to a query, but also include unsuccessful *Request* messages. Therefore, if a consumer has a member nearby, it sends the initial *Request* message. Afterwards, if there was no successful reply, it tries every 2 s, up until 30 s to discover a fitting service. If a request cannot be fulfilled before 30 s we consider the request to be unsuccessful and stop requesting.

The first set of results are shown in Figure 3.7 outlining the discovery latency for different vehicle densities. What we immediately see is the impact of the density on the time it takes until a request is fulfilled. For high densities (i.e., 415 vehicles

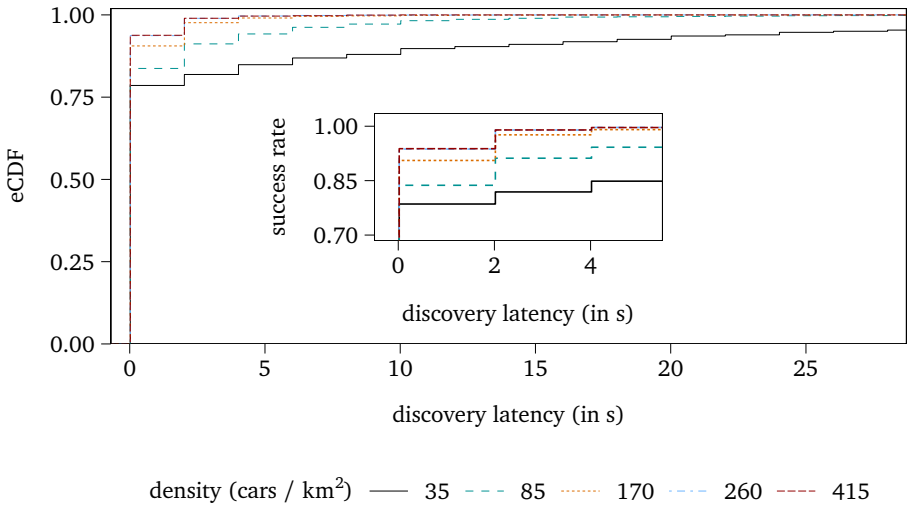


**Figure 3.7** – Discovery latency until a suitable service is found with a service table broadcast interval of 10 s. The different lines represent traffic densities in the range of 35 km . . . 415 km; based on [50] © 2015 IEEE.

per km$^2$) it only takes 2 s to fulfill 99 % requests. This is reduced for lower densities, take for example 85 vehicles per km$^2$ — now only 90 % of requests have a latency of 2 s while it takes 12 s to reach 99 %. This is even worse for the lowest shown density, 35 vehicles per km$^2$. Here, there are multiple instances were 30 s are not enough leading to no successful service discovery. This latency might be acceptable for a M2M scenario with distributed sensing, it certainly is not for user services.

To improve the performance of the lowest traffic density, we opted to reduce the service table broadcast interval (hereafter referred to as only broadcast interval) in steps from 10 s to 0.1 s. We show the results in Figure 3.8. Already a reduction to 5 s shows an improvement regarding the discovery latency. Even better is a broadcast interval of 1 s where the immediate success improves by more than 10 % compared to 10 s. By reducing the broadcast interval even further, no significant improvements were observed. The reason for this is the low vehicle density. If there are not enough members sharing service information, reducing the broadcast interval to aggressively share service tables does not help in acquiring such information. While adapting the broadcast interval can help in high density scenarios to avoid load on the wireless channel, we do not see the need to increase it too much. This is because a moderate broadcast interval works well for low and high densities.

In Figure 3.9 we show the effects of changing the broadcast interval for a traffic density of 170 vehicles per km$^2$. Overall, the delay is even lower. For a broadcast interval of 170 s, 90 % of the requests can be answered immediately. After 2 s, i.e., after sending a second *Request* message, close to 100 % of queries can be answered.

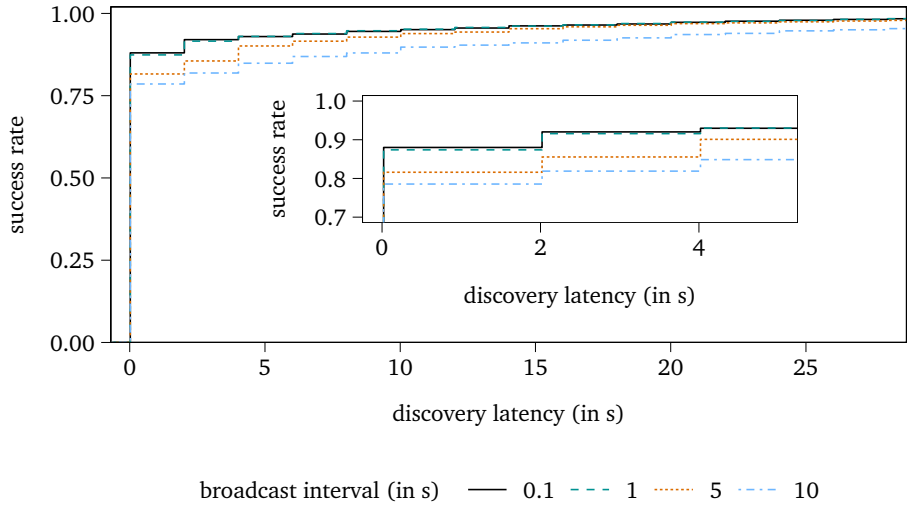


**Figure 3.8** – Discovery latency until a suitable service is found for a traffic densities of 35 vehicles per km$^2$. The different lines represent varying broadcast intervals from 0.1 s to 10 s; based on [50] © 2015 IEEE.

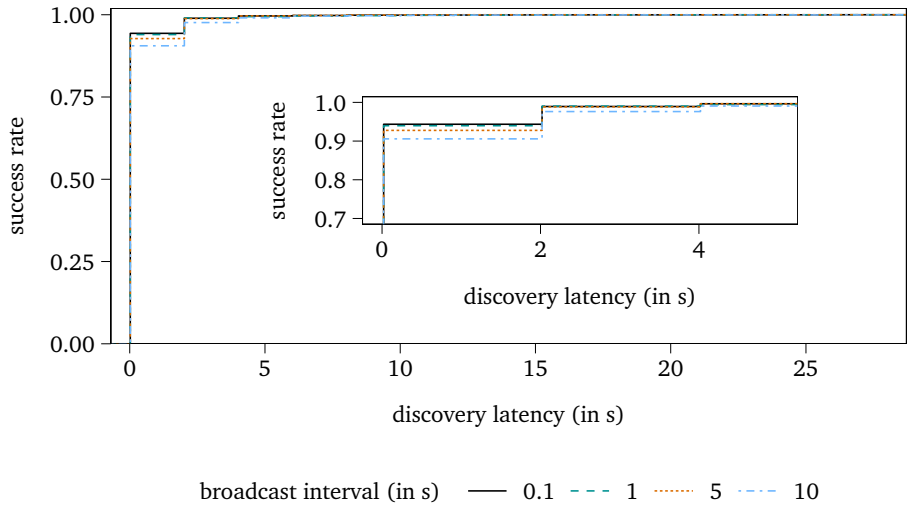


**Figure 3.9** – Discovery latency until a suitable service is found for a traffic densities of 170 vehicles per km$^2$. The different lines represent varying broadcast intervals from 0.1 s to 10 s;

This is true for any broadcast interval, even for the lowest of 10 s. Therefore, if a certain number of Car4ICT members is deployed inside a city, discovery works well if the service tables are proactively shared between members.

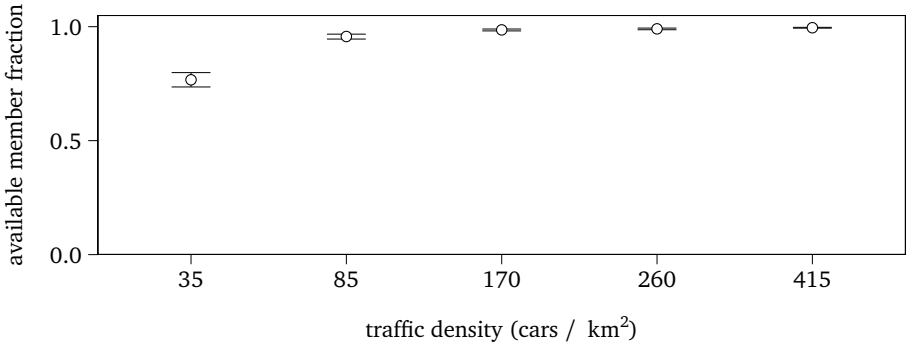


**Figure 3.10** – Availability of a member to the consumer with a broadcast interval of 1 s for different traffic densities. The lines are the 95 % confidence intervals.

Finally, we investigate the availability of members for consumers requesting a service. If no member is available, a consumer is not able to send its request and no delay can be recorded. As we can observe in Figure 3.10, with 35 vehicles per km$^2$, a member is only reachable in 75 % of the cases. For 85 vehicles per km$^2$ this already increases to more than 95 % and for other densities is close to 100 %. Overall, this indicates that a certain number of members need to be driving in a city to provide a user with a functional service. If 25 % of the time, the user cannot connect to the Car4ICT network, the service quality will probably not be satisfactory, independent of the delay if the connection is successful.

## 3.5 Intercity Connections via Car4ICT

One way to extend the number of available services is to use a central service register reachable via cellular networks. As this might not be available or induce additional costs, another way would be to consider searching for services outside of or in other smart cities. By doing this, more services become available for two reasons: (1) more variety of existing services and (2) novel services only available in rural areas or on freeways. In such long-distance scenarios, sharing service tables is not an option, especially for members farther away, as it would make the tables and corresponding sharing messages too large. Therefore, there is the need to search for services after a consumer sends a query to a member. For such queries we mandate to have a location added to direct the search towards a geographic direction. Due to the smaller number of available search directions (due to fewer intersections) in rural areas this can be a valid option. Nevertheless, *Request* messages would travel significantly larger distances compared to urban areas and therefore rely on different routing mechanisms. Consequently, we investigate how effective Contention Based Forwarding (CBF) and Store-Carry-Forward (SCF) are for sending messages over large distances along a freeway.

### 3.5.1 Freeway Service Discovery

Usually in Car4ICT, services are announced by providers and members store this information in their service tables. Members either share such service information by relying on regular broadcasts or actively search for a suitable service when receiving a service discovery *Request*. As we saw in our evaluation in Section 3.4, exchanging service tables allows for a fast service discovery even for low traffic densities. After an appropriate service has been found, data exchange can be done based on information from routing tables or by relying on georouting. To take Car4ICT into rural areas, making more services available and enable intercity connections, we focus on service discovery in such cases. Services can not only be offered locally, but also to potential consumers farther than a few hops away. This increases the number of available services for consumers and provides a higher chance to find the right one. Not only the number of services increases, but also additional types of services might emerge, e.g., routing information for a long-distance journey. Moreover, these changes allow to provide connections to areas without infrastructure, i.e., in the countryside or in an area hit by a disaster. To enable all this, we outline how we adapted Car4ICT with georouting capabilities to work in freeway scenarios.

Exchanging service tables to distribute service information over hundreds of kilometers is not practical as it would create large amounts of data. Therefore, we assume that, for intercity connections, service discovery is triggered by service


**Figure 3.11** – A sketch outlining where to use the different service discovery modes: service table exchanges in cities and CBF in-between them.

discovery queries. If we consider a freeway, there are few junctions where the message flow would not follow the general direction of the freeway. In Figure 3.11 we show a sketch how it would look if Car4ICT spans multiple cities. For service discovery in cities, we rely on exchanging service tables. But for connections between these cities the discovery happens by using a combination of CBF and, if that fails, Store-Carry-Forward (SCF). First, every message is transmitted using CBF. Hereby, every car decides on its own when a message needs to be forwarded. The car which sent the message last, also waits for an acknowledgment and if none is received, switches to SCF. In this mode, the car keeps the message stored and periodically sends it again to find a suitable forwarder. To use both CBF and SCF, messages need to be annotated with a geographic destination. In case of Car4ICT this should be the area in which the consumer roughly expects to find a service, e.g., the next major city along the freeway. Furthermore, a member can send messages with different destinations to potentially increase the number of available services.

Figure 3.12 shows an overview of the used routing algorithm consisting both of CBF and SCF modes. Whenever a car receives a message, it starts in CBF mode. The

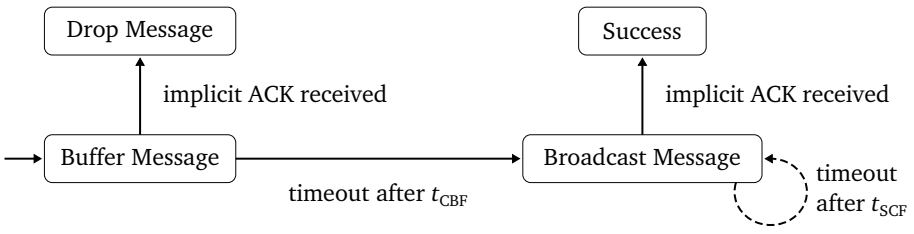


**Figure 3.12** – Basic steps of the freeway Car4ICT routing algorithm after receiving a message. Solid lines are the CBF mode and dashed lines the SCF mode.

car buffers the message for some time $t_{\text{CBF}}$ before broadcasting it. The calculation and parametrization of $t_{\text{CBF}}$ is key to a suitable georouting performance. Initially, the formula outlined by Füßler et al. [127] calculates the buffer times as

$$t_{\text{CBF}} = \begin{cases} t_{\max} \times (1 - \frac{p}{p_{\max}}) & \text{if } 0 \leq p < p_{\max} \\ \infty & \text{otherwise} \end{cases} \tag{3.5}$$

with $t_{\max}$ being the maximal time a message is buffered, $p$ the progress of the packet towards its destination and $p_{\max}$ the assumed maximum transmission distance. Due to $p_{\max}$, messages which get transmitted farther than expected are deleted. This happens even if the message made significant progress towards its destination. The CBF version proposed by the ETSI [131] avoids this issue by using a slightly different formula for $t_{\text{CBF}}$:

$$t_{\text{CBF}} = \begin{cases} t_{\max} + \frac{t_{\min} - t_{\max}}{p_{\max}} \times p & \text{if } 0 \leq p \leq p_{\max} \\ t_{\min} & \text{otherwise} \end{cases}. \tag{3.6}$$

They introduce the parameter $t_{\min}$, the minimal time a message is buffered. If $t_{\min} = 0$, the first case of both versions is similar. But, the second case is different as the ETSI version still sends the message as soon as possible while the one in Equation (3.5) discards the message. By using the ETSI version, messages are not getting lost if they are transmitted farther than expected. Beside all that, we want to point out that it is important to set $p_{\max}$ carefully. Especially setting it too low could lead to colliding messages as cars would retransmit messages far too early.

If a car receives a message and it is not the destination, the car stores the message and calculates the buffer time $t_{\text{CBF}}$ for it. If the message is not received again during this time, the car broadcasts it. All other cars in the surroundings, which now receive the message a second time, stop their timers and drop the message. Therefore, the broadcast acts as an implicit acknowledgment.

If a car sends the message, it starts another timeout $t_{\text{SCF}}$ for SCF. This timeout is canceled too if the car receives an implicit acknowledgment, otherwise it can be assumed that no car closer to the destination received the message. This strongly indicates that there is currently no suitable forwarder towards the destination. Therefore, the car continues buffering the message. Furthermore, each time $t_{\text{SCF}}$ expires, the message is sent again to find potentially suitable forwarders. By doing this, Car4ICT transports the message through areas without any cars. This is crucial on freeways where the network topology is not as dynamic as in urban environments, especially during times with a low traffic density. So, if there is a gap in the number of cars on the freeway, the message is still getting closer to its destination, although at a much smaller pace.

Using the outlined routing modes, it is possible to search for and make use of far distant services. A consumer has two options: (1) perform services discovery as outlined in Section 3.3.3 or (2) keep identifiers after the service discovery. The second option is an extension especially useful for services existing for a longer time, e.g., a fixed sensor along a freeway. By choosing this new option it is possible to immediately use a service — this saves resources, e.g., when regularly acquiring data from such sensors along a freeway. As they are not moving, there is no need to discover them every time.

### 3.5.2 Evaluation

Different services have different requirements when it comes to delay. When we consider weather forecasts, delays in the range of minutes are easily tolerable, but for services like traffic information only shorter delays are acceptable. In the remaining section we evaluate the *discovery delay*, i.e., the round-trip time it takes from sending a request until a suitable service is found. This tells us which services can still be supported when using Car4ICT on a freeway.

We performed the evaluation by simulating a realistic freeway between Tokyo and Osaka in Japan. The used freeway segment is located on the Tomei Expressway connecting Gotemba with Mikkabi and is shown in Figure 3.13. The used map data was gathered from OpenStreetMap[9] and converted by tools accompanying the used road traffic simulator SUMO [103]. Similarly, traffic data was based on real-world

[9] `http://www.openstreetmap.org`

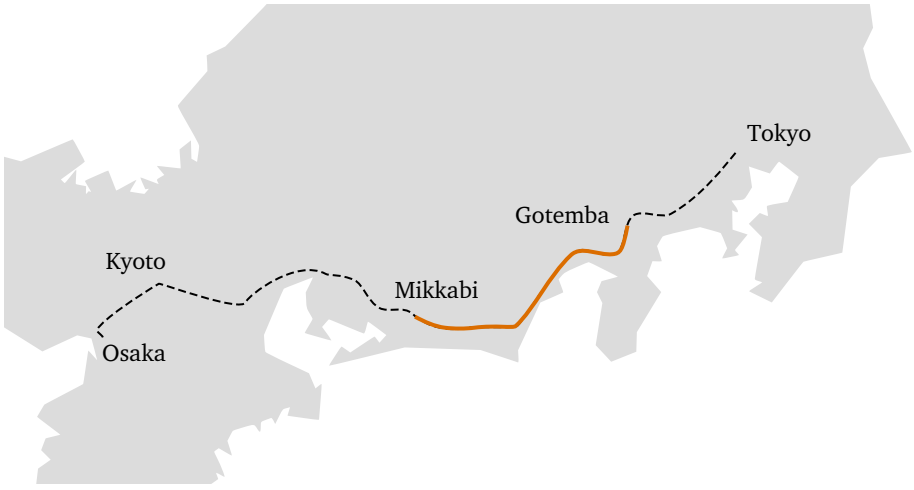


**Figure 3.13** – A map showing the central part of Japan with the Tomei Expressway between Tokyo and Osaka. The highlighted section is the simulated freeway segment between Gotemba and Mikkabi and has a length of roughly 150 km; based on [51] © 2016 IEEE.

traffic data published by the Japanese traffic authorities. Overall, the simulated freeway segment was 150 km long and had mostly two lanes per direction.

To generate the used traffic, we relied on two kinds of traffic statistics provided by the authorities:

- How many cars use an access road [132]. This includes cars entering and leaving the freeway.
- The number of cars driving towards an access road on the freeway [133].

Both pieces of information allowed us to model realistic traffic to be used in our simulations. The densities in both directions over time can be seen in Figure 3.14. In the shown data several spikes are visible due to having more cars driving on access roads compared to the segments between them. On these segments, cars enter and leave the highway in parallel.

We did the simulations during three different hours of the day, each representing a different traffic density. Most traffic happens during the evening rush hour (5pm–6pm), the least traffic is early in the morning (5am–6am), and an average hour of traffic was taken from the afternoon (2pm–3pm).

#### 3.5.2.1 Traffic Generation

To generate traffic out of the available information we required routes in the form of Origin-Destination (O/D) matrices. They contain the number of cars on a road segment with a certain destination. This can be expressed as the number of cars $f_{i \to j}$ entering the freeway at access road $i$ and leaving at $j$.

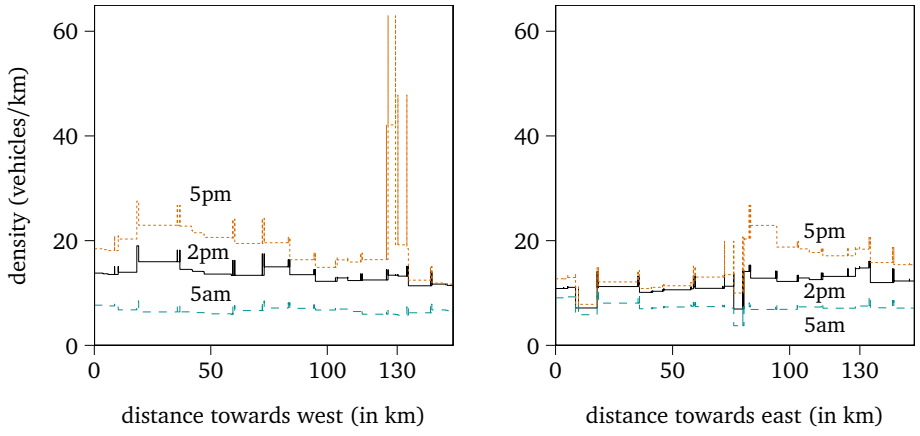


**Figure 3.14** – Overview of the traffic densities for both east and west-bound traffic; based on [51] © 2016 IEEE.

Such matrices were not directly deductible from the provided data, rather we had data gathered from induction loops, which do not distinguish by destination. Therefore, we have two kinds of data for every access road $i$: the number of cars $T_i$ entering or leaving the freeway and $c_i$, the number of cars already on the freeway. As this data does not directly provide O/D matrices, we outline how we estimated them based on what was available to us.

Figure 3.15 demonstrates how we transformed the available data to acquire O/D matrices. First, for every access road $i$ we calculate how many vehicles arrive, $a_i$, and how many exit the freeway, $e_i$:

$$a_0 = c_0, \tag{3.7}$$

$$a_i = \frac{1}{2}\left(c_{i+1} - c_i + T_i\right), \tag{3.8}$$

$$e_i = T_i - a_i. \tag{3.9}$$

These counts allow us to estimate the entries of our O/D matrix by calculating

$$f_{i\to j} = \begin{cases} e_j/c_j \times \left(a_i - \sum_{x=i+1}^{j-1} f_{i\to x}\right) & \text{if } j > i+1 \\ e_j/c_j \times a_i & \text{if } j = i+1. \end{cases} \tag{3.10}$$

Based on these results we generated an O/D matrix for every hour of the day. Using these matrices, we had all necessary information to run our simulations. An outline of the used simulation parameters can be found in Table 3.3. For evaluating the

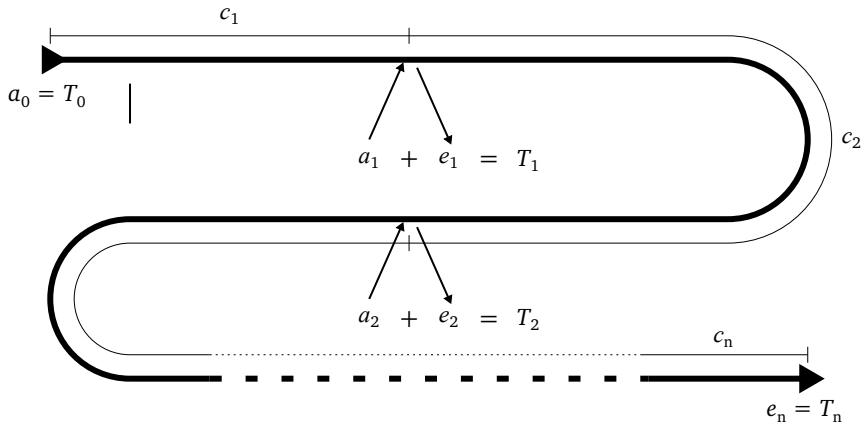


**Figure 3.15** – Outline of the process to acquire the O/D matrices. It is based on the available traffic data, $T_n$ and $c_n$, and produces the used traffic variables, $a_n$ and $e_n$. These are then used to fill the matrix; based on [51] © 2016 IEEE.

performance, we relied on the ETSI version of CBF and investigate the effects of different parametrization and connected car penetration rates.

#### 3.5.2.2 Parameter Study

When discussing the CBF formula we stressed how important it is to select useful values for $p_{max}$. We start our evaluation by investigating this more closely in Figure 3.16 together with the maximum buffer time $t_{max}$. Here, consumer and provider were placed 100 km apart from each. Regarding traffic density, we used the medium traffic density from 2pm to 3pm.

By increasing $t_{max}$ it takes longer to successfully request a service. This is expected as $t_{CBF}$, i.e., the time a message is buffered, depends on $t_{max}$. Similarly, by increasing $p_{max}$, service discovery also takes longer. As already mentioned, if messages cover shorter distances as the maximum assumed transmission distance $p_{max}$, they are sent after buffering them for $t_{min}$ seconds, i.e., after 1 ms. Therefore, every time

**Table 3.3** – Simulation parameters for the freeway simulations.

| Name | Value |
|---|---|
| Simulation duration | 100 s . . . 900 s, 3600 s |
| Distance consumer → provider | 10 km, 30 km, 50 km and 100 km |
| $t_{min}$ | 1 ms |
| $t_{max}$ | 100 ms, 200 ms, 300 ms |
| $p_{max}$ | 500 m, 750 m, 1000 m |
| Penetration rate $r$ | 0.25, 0.5, 0.75, 1 |
| SCF interval | 2 s |
| Physical layer | IEEE 802.11p |
| Max transmission power | 20 mW |
| CBF algorithm | ETSI |

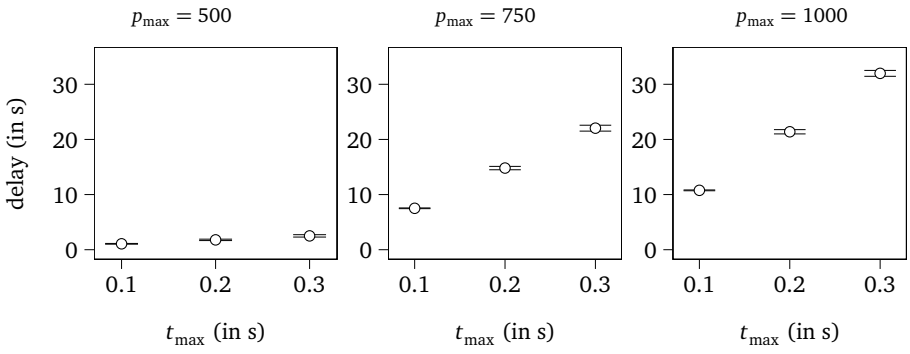


**Figure 3.16** – The parameter study results, showing a delay for a distance of 100 km. Used parameters include $p_{max}$ and $t_{max}$. The error bars represent ± the standard derivation; based on [51] © 2016 IEEE.

such a retransmission happens, the message got already significantly closer to its destination and is sent earlier. The higher $p_{\max}$ gets, the smaller the number of times $t_{\min}$ is used as $t_{\text{CBF}}$. Even worse, messages are buffered longer before being sent.

Before continuing with our evaluation, we investigate the relationship of the buffer time $t_{\text{CBF}}$ and the progress $p$. Overall, there are three different cases:

- $p = 0$ and $t_{\text{CBF}} = t_{\max}$: This rare case indicates that the message made no progress between the last sender and the current car. Nevertheless, the car will try to send it after some time. But it is more likely that another car had a higher $p$ and sent the message earlier.

- $p \in ]0, p_{\max}[$ and $t_{\text{CBF}} \in ]t_{\min}, t_{\max}[$: This will happen regularly as it indicates the message made progress and it is buffered longer then $t_{\min}$.

- $p > p_{\max}$ and $t_{\text{CBF}} = t_{\min}$: If this occurs, the message covered more distance than expected and is sent on as fast as possible. Generally, it is preferred that this happens but only for a small number of cars in parallel as otherwise it would trigger collisions.

The ETSI proposes a value of 1000 m for $p_{\max}$ [131]. As most messages in our simulation covered a distance between 400 m and 500 m we argue that it makes more sense to use $p_{\max}$ with a value of 500 m. By doing this, the discovery latency is reduced to very low delays and provides reasonable performance for most of the envisioned Car4ICT services. To summarize, we can see that if all cars are equipped with a Car4ICT module, providing low discovery delays for services is possible. This holds true for most kinds of such services, except for safety related ones.

#### 3.5.2.3 Penetration Rate Study

Initially, the penetration rate of deployed connected cars will be low. Therefore, it is interesting to investigate the effects of a lower penetration rate on the performance of long-distance Car4ICT service discovery. For this last set of results, we reduced the rate of cars equipped with a Car4ICT module from 100 % step wise to 25 %.

Beside reducing the penetration rate, we also investigated the *discovery delay* over four different distances: 10 km, 30 km, 50 km, and 100 km. If a car just ferries the data, it would roughly take 1 h to cover the maximum distance. Therefore, we limited the simulation time to 3600 s because after this time we would be faster without relying on wireless connections. Furthermore, this is the upper bound for maximum *service discovery delay*.

We present the results of this evaluation in Figure 3.17. For a penetration rate of 1.0, i.e., all cars are equipped with a Car4ICT module, the delays are rather short, at maximum 12 s to cover 100 km. We observed similar results for a penetration rate

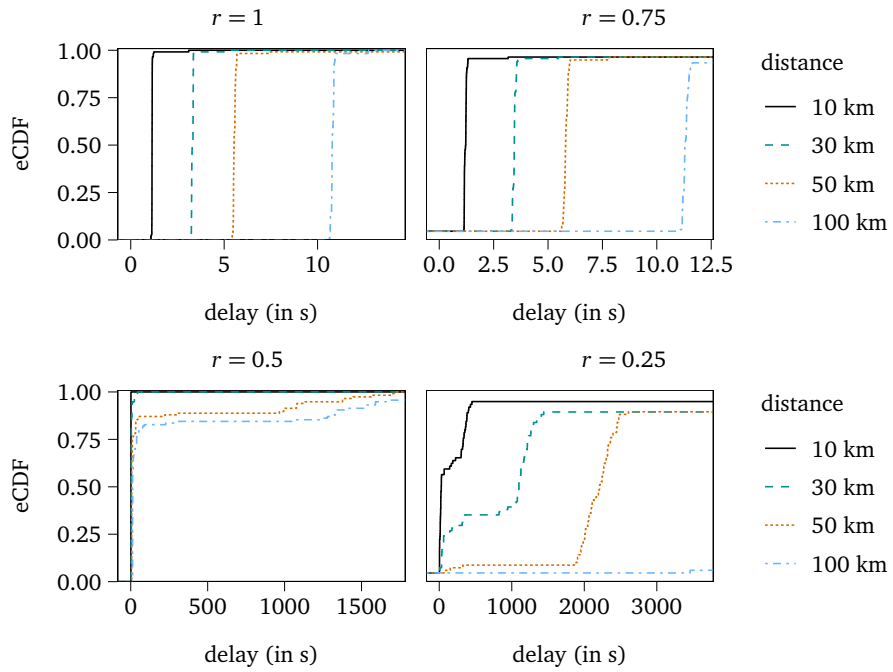


**Figure 3.17** – The effect of the penetration rate $r$ onto the delay for distances in the range of 10 km . . . 100 km; based on [51] © 2016 IEEE.

of 75 %, the delays are only slightly longer due to a reduced number of available forwarding cars. If only half of the cars are Car4ICT members, i.e., $r = 0.5$, most discoveries are still happening very fast. Nevertheless, for the larger distances, i.e., 50 km or 100 km, larger delays of up to 1500 s occur. This is because the number of potential forwarders decreases even further.

Finally, for a penetration rate of 25 %, only the shortest distance of 10 km provides useful results. Nevertheless, nearly all messages arrive for 30 km and 50 km during the observed time, but not for 100 km. Here we observe an issue where the simple SCF algorithm rarely selected cars traveling in the opposite direction essentially moving the message backwards. A more sophisticated algorithm could factor in the travel direction to avoid such situations. But, they do not occur with a higher penetration rate as a car moving in the wrong direction is still able to use CBF to successfully forward the message. Overall, we can see that lower penetration rates start to influence the performance when they fall below 50 %, which leads to useless services. With higher penetration rates, serviceable performance is still possible.

## 3.6 Lessons Learned

In this chapter we proposed a service discovery architecture named *Car4ICT*. It utilizes connected cars as building blocks for future ICT systems in smart cities and beyond. One concept of smart cities is the generation, collection, and processing of sensor data. Many additional data sources are envisioned to be deployed and current technology will not be able to deal with the resulting data flow [37]–[39]. One solution would be to wait for 5G, but deployment of such a new technology is expensive and might not solve all issues. Our approach is to instead use connected cars, which, beside communication capabilities, also provide storage and processing resources. And, to make it even better, they are not a futuristic concept.

Our proposed Car4ICT architecture assists users, both humans and machines, in offering and utilizing various kinds of services, e.g., storage offloading, data collection, or multimedia file sharing. In our evaluation, we were able to show that it is possible to discover services in a relatively short amount of time. Nevertheless, we also stumbled upon a well-known issue in vehicular networks, namely the penetration rate of equipped vehicles [75]. Initially, there will not be many vehicles available, which seems to limit the performance of Car4ICT. A low penetration rate led to worse service discovery, only 75 % of the queries were answered fast, others observed a long delay. Adapting the parametrization improved the performance, but still leaves room for improvement. Beside improving the algorithm, a future research direction might be to incorporate existing infrastructure, e.g., cellular 4G/5G networks, during the introductory phase of connected vehicles.

Beside urban environments, we also considered long-distance connections in rural areas, i.e., service discovery on freeways. We had to adapt our service discovery approach to be location-based and use Contention Based Forwarding (CBF), a receiver-based routing algorithm. Initially developed by Füßler et al. [127], a modified version of CBF is nowadays part of the ETSI ITS-G5 standard [131]. Our real-world simulation shows that indeed CBF works well on freeways, but also that correct parametrization is important. Using the configuration the ETSI standard recommends led to significantly larger delays compared to lower parameters. This opens further research opportunities regarding CBF, e.g., develop an adaptive version where, if needed, a more aggressive parametrization is used. Furthermore, it might be of interest to perform the parameter study we did in simulation in real-world experiments and get more realistic insights. Finally, we also confirmed that the penetration rate is a deciding factor in our freeway scenarios. While the performance was sufficient down to 50 % of equipped vehicles, lower rates could lead to long delays. Parts of it can be attributed to our CBF implementation and disregarding the driving direction. But, like the urban scenario, it might make sense to rely on any kind of cellular infrastructure (e.g., IEEE 802.11p Roadside Units or LTE eNodeBs)

during the introductory phase where a low penetration rate of equipped vehicles is to be expected.

Generally, it would be interesting to move the Car4ICT concept away from single cars. This could be done by replacing Car4ICT members, the core nodes of the network, with clusters of cars instead of single ones. In the next chapter, we partly do this by having a cluster of parked cars act as a single node. By performing this step, Car4ICT could be easily integrated into the Micro-Macro Cloud concept with every Car4ICT member essentially being a micro cloud.

# Chapter 4

# Vehicular Micro Clouds in Action

THIS chapter presents two micro cloud architectures. Being part of the Micro-Macro Cloud (MMC) concept presented in Chapter 2, micro clouds are small-scale clusters consisting of proximate vehicles envisioned to work together in order to perform tasks, offload resources from a larger macro cloud, and provide services [36]. Depending on the cloud's mobility, there are several ways how to form them. We focus on stationary micro clouds formed using parked (Section 4.3) and moving cars (Section 4.4). Furthermore, we investigate the feasibility of these clouds by proposing services running on top of them. These services include providing virtual network infrastructure, collecting information, and preserving locally relevant data to offload a macro cloud.

With these capacities, i.e., service provision and resource offloading, micro clouds are an example how connected cars enable smart cities. The two architectures presented in this chapter relate to two of the research question categories outlined in Chapter 1: *Virtual Network Infrastructure* and *Micro Clouds and their Data*. Research presented in this chapter is based on the following publications:

- F. Hagenauer, C. Sommer, T. Higuchi, O. Altintas, and F. Dressler, "Using Clusters of Parked Cars as Virtual Vehicular Network Infrastructure," in *8th IEEE Vehicular Networking Conference (VNC 2016), Poster Session*, Columbus, OH: IEEE, Dec. 2016, pp. 126–127
  My contributions to this conference paper were taking part in the development of the proposed virtual network infrastructure. To make the concept a reality, I participated in the preparation of the outlined central research questions.

- F. Hagenauer, C. Sommer, T. Higuchi, O. Altintas, and F. Dressler, "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)*, Snowbird, UT: ACM, Oct. 2017, pp. 57–64
  My contributions to this conference paper were the design, development, and evaluation of the proposed micro cloud algorithm. Based on the idea that parked cars can be used as virtual network infrastructure, I developed a solution using the Virtual Cord Protocol. Hereby, I focused on a core part of the architecture, namely gateway selection. Such a selection lowers the number of nodes a passing car can connect to. This selection barely reduces the performance of applications exploiting the virtual infrastructure. Finally, I evaluated the architecture in varying circumstances with a focus on the influence of the gateway selection both in an artificial and a realistic scenario.

- F. Hagenauer, C. Sommer, T. Higuchi, O. Altintas, and F. Dressler, "Vehicular Micro Cloud in Action: On Gateway Selection and Gateway Handovers," *Else-*

*vier Ad Hoc Networks*, vol. 78, pp. 73–83, Sep. 2018
My contributions to this journal article were the improvement and evaluation of the extended virtual network infrastructure algorithm. I focused on developing the handover part of the algorithm enabling longer connectivity. By performing handover, cars can transfer more data or use complex protocols. Eventually, I evaluated the algorithm in two realistic scenarios and focused on the performance of the handover improvement.

- F. Hagenauer, C. Sommer, T. Higuchi, O. Altintas, and F. 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)*, Snowbird, UT: ACM, Oct. 2017, pp. 31–35
  My contributions to this conference paper were the design and evaluation of an algorithm forming micro clouds at geographic relevant locations, i.e., a map-based approach. I developed the idea where micro clouds are formed using clustering concepts at specific locations coordinated by Access Points. The concept was evaluated in a basic artificial scenario with a single application, data collection from Cluster Members. Furthermore, the evaluation was not only done using IEEE 802.11p, but also in combination with LTE.

- F. Hagenauer, T. Higuchi, O. Altintas, and F. Dressler, "Efficient Data Handling in Vehicular Micro Clouds," *Elsevier Ad Hoc Networks*, vol. 91, p. 101 871, Aug. 2019
  My contributions to this journal article were the development, extension, and evaluation of the previously proposed geographic clustering algorithm. In addition to the data collection service, I took part in the development of a further service, which keeps local data in the micro cloud. For this service, I developed improvements increasing the performance significantly. Finally, I evaluated these findings in a realistic Manhattan grid scenario for varying traffic densities showing how the improvements affect the performance.

## 4.1 Motivation

Micro clouds as part of the Micro-Macro Cloud (MMC) concept offer diverse opportunities. Besides providing access to processing, storage, and communication resources, they are also enabling numerous novel services [36]. Examples include virtual infrastructure, driver-assistance services providing traffic information, and sharing multimedia data. One appealing reason for micro clouds is their flexibility. They can be formed using moving or parked cars [45], by relying on existing clustering solutions for vehicular networks [44], or develop completely new concepts.

In this chapter we provide a glimpse into the active MMC research field by outlining two concrete examples for the formation of micro clouds. First, we discuss how to provide virtual networking infrastructure based on micro clouds consisting of parked cars. Second, by exploiting moving cars, we explore a micro cloud formed at geographic locations and its data management services.

### 4.1.1 Parked Cars Providing Virtual Infrastructure

Different vehicular networking applications have varying requirements regarding delays, throughput, or connectivity [19], [21]. While safety applications do not work with large delays, other types, e.g., infotainment or efficiency applications, can cope with them. When it comes to throughput, the picture is similar: safety applications usually work only with small amounts of data. But, both efficiency and infotainment applications might require more data. Already complex traffic information can accumulate a few kilobytes and transferring multimedia data necessitates even more data, e.g., an image taken by a camera with multiple MB.

If we now consider transmitting larger amounts of data, sender and receiver need to be connected for a long time. Considering the dynamic nature of vehicular networks in urban environments, this certainly cannot be guaranteed. Even with full market penetration, the topology of a vehicular network consisting of driving vehicles is only connected during rush hours [88]. During other times of the day, the network is heavily fragmented and does not provide continuous connectivity. The reasons for this are twofold: (1) buildings block transmissions even between cars close-by and (2) the traffic conditions vary extremely, especially in suburban areas.

As outlined in Section 2.4, to achieve a better-connected network, and in turn improve longer transmissions, roadside infrastructure can be incorporated. Potential candidates include Wi-Fi Access Points (APs), IEEE 802.11p Roadside Units (RSUs), or cellular base stations [20], [91] (e.g., eNodeBs (eNBs)). If such infrastructure nodes are installed at beneficial locations, the overall performance can be improved, e.g., by improving routing performance [86]. They can coordinate network traffic in the surroundings to avoid congestion and are able to work as relay nodes improving

the connectivity [86]. If APs are connected to each other, to a macro cloud, or to the internet, it is possible to send data over large distances. But, deploying infrastructure incurs additional costs and, after being built, requires money to operate.

Moreover, building infrastructure does not necessarily solve the issue of coverage. In many western countries cities are filled with Wi-Fi APs and eNBs — still, the coverage might be not enough to provide continuous data connections [41]. Buildings in cities are blocking radio transmissions leading to bad performance for certain nodes [43]. Regarding Wi-Fi APs, the contact time between a car driving by and the AP is the limiting factor. If the transfer requires complex communication protocols or the transmission of large files, the contact time needs to be long. This can be mitigated by splitting the transmission into multiple fragments and sending them via subsequent connections. Unfortunately, this solution leads to larger delays for a complete file transmission, especially if no handover between APs is possible.

As a potential solution to the outlined issues, we expand the idea of clustering parked cars [53], [54] to establish additional virtual network infrastructure. Our proposed architecture does not require any existing infrastructure and is therefore completely distributed and independent of any cellular provider. Nevertheless, if there exists infrastructure, it can be integrated seamlessly. In the process of developing the architecture, we were able to answer some of our previously stated research questions [53] (cf. Section 1.3). Among other things, the research questions ask how to form such a micro cloud providing virtual infrastructure, how to connect to it without overloading the channel, and how to enable longer lasting data sessions. The micro clouds presented in Section 4.3 support handover for longer transmission and refinements to reduce the channel load from control messages.

### 4.1.2 Micro Clouds and Data Management

Beside virtual infrastructure based on parked cars, we also investigate data management in vehicular micro clouds. Hereby, in the second part of this chapter (Section 4.4), we focus on moving cars and how it is possible to form stationary micro clouds at specific geographic locations. To group cars, i.e., cluster them, we do not rely on cars being close to each other, but rather on them being close to a specific geographic location. Such clusters exist for a longer time, but cars are not part of it for long periods of time — except if they have to wait at an intersection. For these clouds, we investigate two ways of data management showing how they are able to provide the foundation for services.

The clustering objective is to select Cluster Heads (CHs) close to locations providing connectivity into as many directions as possible. For example, by placing CHs in the middle of intersections we utilize line-of-sight communication to Cluster Members (CMs) as they approach the intersections using streets leading towards it.

To select a CH and its CMs, i.e., to compute the cluster configuration, we make use of existing infrastructure, e.g., a Wireless LAN (WLAN) AP or an LTE eNB. APs acquire the data necessary for cluster coordination from periodic beacons sent by cars in the vicinity. By placing these infrastructure nodes close to relevant geographic locations, they receive data from vehicles in a position to be considered for the cluster. Moreover, the APs might be connected to a backbone providing a direct connection to a macro cloud or the internet.

Beside a concept for forming a cluster, we outline two micro cloud data management services. These services should be a base for building user-facing applications on top of them. The two discussed services are:

- ***Collect* Service**: This service periodically collects data from all CMs and uploads it to the macro cloud. At a fixed interval, every CM sends its current information to the CH. Prior to uploading the data, the CH processes and aggregates the data. This is done to offload resources in two ways: (1) to offload the macro cloud (less processing needed) and (2) to reduce load on the wireless channel (less data to be transferred).

- ***Preserve* Service**: This service preserves data in a cluster without uploading it to the macro cloud, i.e., offloading storage. Several services provided by vehicular networks rely on locally relevant data and therefore, there is no need for uploading. To keep the data in a cluster it needs to be replicated as CMs change rapidly because of the dynamic nature of vehicular networks.

## Contributions

This chapter discusses two kinds of micro clouds. First, in Section 4.3, we present an architecture for micro clouds consisting of parked cars. The focus is not in the formation of these clouds, but rather on two features increasing their potential: *gateway selection* and *handover*. Second, in Section 4.4 we outline the second micro cloud architecture — this time built upon moving cars. Beside describing the process of forming such a cloud, we present two micro cloud services: data collection and preserving data. The main contributions of this chapter are:

- We describe how to form and maintain micro clouds consisting of parked cars. For this, our solution is based on existing clustering concepts. As an application scenario we consider these micro clouds to provide virtual infrastructure. Furthermore, we propose two improvements to these clouds: (1) using *gateway selection* to reduce the load on the wireless channel and (2) a *handover mechanism* to extend the connection time of a passing car to the micro cloud. For this, we select cars from the cloud perimeter to be gateways, which are then accessible from driving cars. By doing this, we aim to reduce

the load on the wireless channel while still providing the same service quality to users. To access a micro cloud via multiple gateways, a user must perform a handover between them. Therefore, it is possible to transmit more data to the cloud and make use of more complex protocols.

- We describe a concept to form micro clouds at geographic locations beneficial for in-cluster communication. Suitable locations are for example intersections with many streets leading towards them. We do this by relying on a centralized clustering mechanism using holistic information to select CHs and CMs. Furthermore, we propose two services outlining the potential of data management in such micro clouds. By fulfilling two different demands (collecting data from the cluster and keeping data inside the cluster), we are able to demonstrate the versatility of micro clouds as part of the MMC concept.

## 4.2 Preliminaries

Both micro cloud concepts and their respective applications rely on contributions from different research areas related to vehicular networking. This includes, but is not limited to, RSUs, clustering, gateway selection, handover mechanisms, Vehicular Clouds (VCs), and Mobile Edge Computing (MEC). First, we discuss the MMC concept and where our proposed approaches fit in. Second, we present existing approaches for virtual infrastructure in vehicular networks. This is followed by an overview of relevant publications considering clustering and potential applications. Finally, we outline works from the area of gateway selection and handover approaches in vehicular networks.

### 4.2.1 Micro and Macro Clouds

Both micro clouds presented in this chapter can be considered part of the MMC concept outlined in Chapter 2. These micro clouds, beside offloading processing and

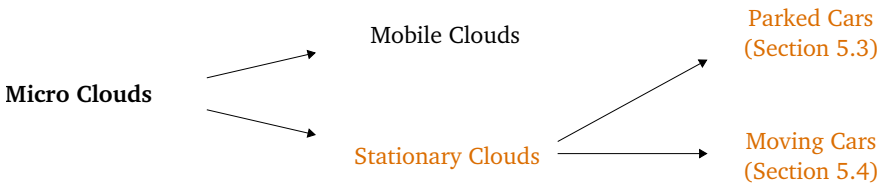


**Figure 4.1** – An overview of the multiple micro cloud forms. This overview is based on the taxonomy discussed by Higuchi, Dressler, and Altintas [45]. Highlighted are the two stationary micro cloud concepts from this chapter.

storage resources (like Mobile Edge Computing), are also the baseline for offering numerous new services (like Vehicular Clouds) [36]. In Figure 4.1 (extended version from Higuchi, Dressler, and Altintas [45]) we show various types of a vehicular micro cloud: micro clouds can either be moving or stationary. A moving cloud travels through a city while a stationary cloud is formed at a fixed location and does not move. Furthermore, a stationary micro cloud does not necessarily rely on parked cars (Section 4.3), but can also be formed using moving cars (Section 4.4).

### 4.2.2 Towards Virtual Networking Infrastructure

By placing RSUs, the stability of vehicular networks is believed to improve. One example of RSUs improving the performance of a vehicular network is *ROAMER* [86]. It is a routing protocol exploiting roadside infrastructure to connect distant cars with each other while using basic broadcast for cars close-by. Nevertheless, the authors state that there are still certain issues like the ones mentioned by Naboulsi and Fiore [88] especially in scenarios with low traffic densities.

If no infrastructure is available, it is certainly possible to rely on parked connected cars. How this works for a safety application has been outlined by Sommer, Eckhoff, and Dressler [90]. The authors exploit parked cars to warn driving cars at junctions. This is done by parked cars relaying safety messages from moving cars to others. As these parked cars potentially reach more cars than the sender, the reliability of such warnings is improved.

Virtual infrastructure does not only improve safety applications, but can also be used for infotainment as outlined by Malandrino et al. [30]. In their algorithm, parked cars assist an RSU in streaming data. The cars download videos from the internet and share them with passing cars. By applying various optimizations, cars are able to improve the algorithm's performance.

Overall, nearly all kind of applications benefit from available infrastructure, e.g., an RSU or parked cars providing virtual infrastructure.

### 4.2.3 Clustering

To be able to manage vehicular networks in general, clustering is often considered a key concept. Clustering helps to organize a potentially large number of cars in a hierarchical structure [44], [134]. This is further backed by Sucasas et al. [135], who survey general clustering concepts for cooperative wireless networks and note that vehicular networking is one of the areas with the most potential. According to the authors, a lot can be improved, especially regarding quality of service and how to handle scenarios with high mobility. Basically, a micro cloud is often formed using clustering techniques.

The basic idea of clustering is to form groups of cars to cooperatively perform tasks acting basically as a single node. Cooper et al. [44] survey clustering protocols in vehicular networks and describe the core principles needed to form a cluster. In most of the approaches, cars are grouped based on similar metrics, e.g., speed, direction, geographic position, neighborhood, or data interest. This is based on an approach for Mobile Ad Hoc Networks (MANETs) [136], which was later adopted to be used in Vehicular Ad Hoc Networks (VANETs) [137]. But, one of the main metrics for clustering in MANETs was the energy consumption. As this is not an issue in vehicular networks, a wide range of algorithms developed for MANETs is optimized in the wrong direction [42].

Cluster coordination is usually done by a distinct member of the cluster, the Cluster Head [44]. It has to maintain the cluster and coordinate other members. Nevertheless, it might not be the best selection when it comes to communicating with cars not being part of the cluster or other clusters, i.e., acting as a gateway.

Beside using a combination of metrics, there exist approaches relying on different ideas. The use of evolutionary algorithms is proposed by Cheng et al. [138]. Another example by Zahidi et al. [139] is using Integer Linear Programming. Nevertheless, most of these concepts are theoretical approaches with several assumptions like fixed communication range or stationary cars. Such assumptions makes it difficult to apply them to realistic conditions.

Cooper et al. [44] outline the steps used by most clustering algorithms, which we also follow with the concepts presented in this chapter. The four stages are:

1. A coordinator node (or multiple nodes in a distributed system) gathers control data from prospective CMs.

2. Based on the control data, the cluster configuration is computed. One node becomes CH, the others CMs.

3. The computed configuration is shared with all CMs and the CH.

4. Upon CMs receiving this information, the cluster itself becomes operational and is now able to perform tasks, e.g., collect data, or, even more generally, provide services.

Most existing clustering algorithms are domain specific, and vehicular networks are no different. Thus, different solutions have been designed for use in freeway [140]–[144] and urban [45], [80], [145]–[147] environments, respectively. While the traffic patterns on freeways are rather simple (few chances to enter or exit), urban environments are more dynamic [88]. This results in network fragmentation so that clustering algorithms might rely on additional infrastructure (e.g., RSUs or cellular base stations) for coordination [146], [148]. Nevertheless, intersections and

other landmarks are helpful for efficient clustering [87], [147] as cars close to these landmarks may further experience better connectivity.

Clustering algorithms are either developed for a specific application domain, focus on generic packet delivery, or try to maintain a cluster for as long as possible. Looking at clustering approaches for vehicular networks, several focus on safety applications [140], [145], [149]) while others take a more generic approach [143], [146], [150], [151]. Safety applications usually try to send warning messages to cars nearby (urban) [145], [149] or specifically to following cars (freeway) [140]. More generic clustering algorithms do not rely on a certain application, but focus on generating long-living cluster. Wang and Lin [143] proposed 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, Bakht, and Kravets [150] proposed QuickSilver, an algorithm to support all kinds of applications by combining node-centric and content-centric communication. Similarly, Caballeros Morales, Hong, and Bang [146] developed a clustering solution based on the cars' destination leading to clusters traveling through the city instead of staying at a fixed location.

### 4.2.4 Gateway Selection

To reduce the load on the wireless channel, our micro cloud based on parked cars relies on gateways. Generally speaking, gateways are cluster nodes communicating with other clusters or are even part of multiple clusters [44], [135]. Such gateways are not necessarily the CHs as they might be selected based on different, potentially application-specific, metrics. For our micro clouds, gateways act similar by being CMs via which passing cars connect to the cluster. The best selection of such gateways is a minimal set of cars so that it enables seamless connections to cars not being part of the cluster. This is like *k-barrier coverage* in ad-hoc networks [152]. The idea is to cover a border (e.g., of a country to be kept under surveillance) by at least $k$ nodes. This does not only cover surveillance, but also routing as outlined by Dai and Wu [153]. Using neighborhood information, the authors select certain gateways. Similarly, our gateway selection mechanism relies on geographic positions to provide enough communication coverage.

### 4.2.5 Handover Management

If a driving car wants to transmit large files to an RSU, it probably is not able to do so by connecting to a single unit. It will need to connect to multiple RSUs it passes by to extend the connection time. Similarly, if a car connects to the parked car micro cloud we propose, it needs to connect to several gateways while passing by. This is achieved by enabling horizontal handover, i.e., seamlessly switching gateways.

Another approach to maintain a connection would be to perform a vertical handover, i.e., switching the used technology to use a better suited one [20]. But, more generally, as discussed by Ghosh et al. [154] and Bali, Kumar, and Rodrigues [155], research in vehicular networks usually does not consider any kind of handover. The authors claim that having an efficient handover mechanism is one of the core research issues to enable efficient clustering.

One of the few handover mechanisms for vehicular networks was outlined by Ghosh et al. [156]. Their idea relies on probabilistic handovers between RSUs based on transmission regions. Another approach was developed by Huang, Chiang, and Hsu [157] to support a packet forwarding algorithm. By building a hierarchy, the authors let a common ancestor decide which one of the selected candidates is the most suitable. The assumption, that all potential candidates are connected to each other, is not suitable for dynamic vehicular networks, especially if they are moving. Furthermore, to the best of our knowledge, the authors only analyzed their scheme using analytical methods, and did not explore the effect of more dynamic scenarios.

Mouton et al. [158] discuss another handover scheme inspired by clustering. Their approach relies on a variety of parameters similar to clustering algorithms, i.e., direction, the road the car is on, and deployed network information. Unfortunately, their scheme necessitates the extension of the network stack with multiple modules making it hard to integrate into existing architectures.

### 4.2.6 Data Management Services

We propose two concrete data-management services for our micro clouds. Therefore, in the following, we discuss literature related to these two specific use cases. The first service *collects* data from CMs and uploads it via the macro cloud to a data center. Data collected by micro clouds can be a basic resource for a wide range of services, e.g., safety applications [159] or Traffic Information Systems (TISs) [75]. Such a system was proposed by Raya, Aziz, and Hubaux [159] and supports secure data collection, not reducing channel load. Channel load was considered by Ibrahim and Weigle [160], who outlined a TIS capable of aggregating data even with large amounts of road traffic. Rémy et al. [79] propose a centralized system where an eNB takes care of collecting data from CHs. In their approach it is not exactly clear how clusters are organized (which CM is part of which cluster, which cars are selected as CHs), but they rather focus on maintaining the cluster and data collection. Their basic approach also follows the core steps outlined by Cooper et al. [44]. A more recent example is provided by Taherkhani and Pierre [145] who base their safety message distribution on previously collected data from driving cars. To make this as efficient as possible, the authors provide systems to detect a congested channel, how to manage the cluster and in turn limit the detected congestion.

The second service *preserves* data inside the cluster, essentially offloading resources from the macro cloud. Many potential services rely on locally available data [21], [32]. Therefore, it makes sense to keep such information available and even refrain from uploading it to the macro cloud. This has the potential to reduce the delay for requests and offload resources from the macro cloud, i.e., directly facilitating MEC [34], [96].

## 4.3 Micro Clouds with Parked Cars

We propose to create virtual networking infrastructure by exploiting clusters of parked cars, i.e., by forming micro clouds. Such infrastructure can subsequently be used by driving cars to use provided services and improve connectivity. Available services could be offloading processing, or storage resources, or providing connectivity to a central server in the back end. Furthermore, the infrastructure can be used to coordinate the nearby network to avoid load on the channel. In Figure 4.2 we highlight which kind of micro clouds is considered in this section, i.e., stationary micro clouds consisting of parked cars (B and D).

The underlying idea is shown in Figure 4.3 where a car passes by clusters of parked cars. At the beginning the car connects to the left cluster, starts a data transfer, and continues its journey. After some time, the car connects to the other cluster and starts a second transfer. While passing both clusters, the car continues to exchange data with both clusters, connecting to different CMs over time. Nevertheless, each of the clusters acts as a single entity.

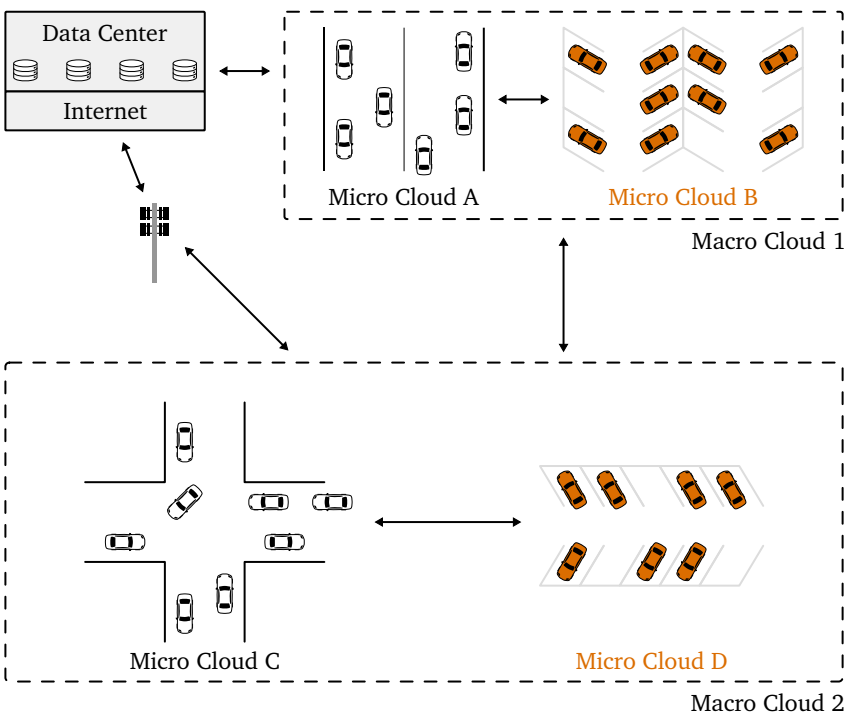


**Figure 4.2** – Overview of the MMC architecture from Chapter 2 where the micro clouds considered in this chapter are highlighted. Both micro clouds B and D consist of parked cars.

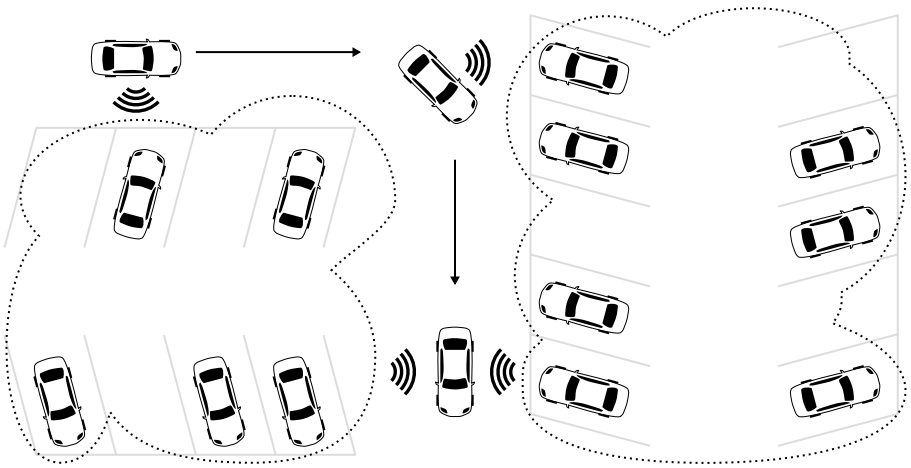

**Figure 4.3** – Outline of the micro cloud concept relying on parked cars. A driving car passes two clouds, each of which forms the own virtual network infrastructure. The gateways hand over the connection to provide longer connectivity to the passing car; based on [55] © Elsevier B.V.

We do not only cover the formation and maintenance of the cluster but also propose two enhancements:

- Not all cars in the cluster communicate to passing cars. This is done only by a subset, denoted as *gateways*. By selecting them, we aim to reduce the load on the wireless channel.
- After selecting gateways, *handover* between them is even more important. Due to the lower number of available cars to connect to, a seamless handover enables sufficient performance while being connected to the micro cloud.

### 4.3.1 Requirements

While we do not demand unusual technologies, there are still multiple requirements for the involved cars. Most of them are considered to be part of any connected car:

- **Vehicle-to-Vehicle Communication**: Parked cars should be able to communicate with each other using a wireless networking technology. We recommend to equip cars with short-range technology like IEEE 802.11p, Wi-Fi, or LTE-D2D because they do not necessitate a coordinator node.
- **In-Cluster Connectivity**: Parked cars should be able to send messages to all cluster members using the above-mentioned networking technology. This can be done by direct communication or by using multi-hop communication to cover larger distances. Nevertheless, a small delay is preferable to enable fast

handovers between gateways. There is no specific routing protocol necessary as cars do not move and therefore protocols developed for ad-hoc networks can be used (in our case Virtual Cord Protocol (VCP) [161]).

- **Geographic Positioning**: Parked cars should be able to determine their geographic location by using a Global Navigation Satellite System (GNSS) like GPS, GLONASS, or Galileo. This necessary to perform handover operations and select the best suited cars as gateways.
- **Local storage**: Parked cars should be able to store control information in a local Knowledge Base (KB). Such information is needed to perform handovers and select the best suited gateway. In our concrete implementation, we rely on VCP. It does not only provide routing but also acts as a Distributed Hash Table (DHT) [161].

### 4.3.2 Clustering Algorithm

To form the micro clouds, we rely on established clustering concepts, which usually necessitate three cluster operations. These steps are similar to the ones outlined by Cooper et al. [44], although, looking at the process from a different angle. The *Setup/Gather Control Data* phase takes care of integrating a car into a cluster either by joining an existing one or setting up a new one. Usually the longest phase is the *Maintenance/Calculate Cluster Configuration* phase which takes care of keeping the cluster operational. During this phase, cars exchange necessary control information to keep the cluster running. The last phase, i.e., the *Departure* phase, takes care of cars leaving the cluster and dissolving the cluster if applicable. Note that, we focus on the first two phases in this thesis as they are more challenging due to higher dynamics. We outline the third stage to complete the algorithm.

- **Setup Phase**: As the proposed cluster consists of parked cars only, a car needs to stop in order to participate. After it is parked, it listens to *cluster beacons*. They are transmitted repeatedly for two purposes: inform potential new CMs of the cluster and exchange control information necessary for maintenance. If the newly parked car has not received any such beacons it initiates a new cluster. Otherwise, it notifies the existing cluster of its presence and joins it. To do this, the car sends its current position to a CM, which then forwards it to the coordinator node in charge of selecting gateways. For an improved gateway selection, we recommend that cars do not join clusters across the street. This leaves us with two kinds of clusters: (1) clusters formed along a street and (2) clusters in a parking lot. If created along the street, a cluster takes the form of a one-dimensional curve. A cluster in a parking lot covers a two-dimensional area.

- **Maintenance Phase**: As long as the cluster exists, cars regularly share *cluster beacons*. Most importantly, they enable to detect if a car left the cluster and consequently a network link within the cluster is broken. Furthermore, they allow potential new CMs to discover the cluster. Additionally, gateways may amend these beacons with information indicating how to connect to the cluster. Cars passing by receive these messages and can connect to the cluster in order to use provided services.

- **Departure Phase**: There are two ways the last phase is triggered: either by a car leaving the cluster or if a broken link is detected. If a car signals that it is leaving the cluster (e.g., by detecting a key in the ignition), it informs the cluster about it. Afterwards it sends all data stored in the local KB to its neighbors according to the updated DHT. If the underlying DHT supports replication of data this step might not be necessary. After this step is completed, all CMs take care of deleting now invalid control information. Note that, a car leaving a parking lot takes at least multiple seconds — enough time to send large amounts of data to other CMs.

In our concrete implementation of the proposed architecture we use the VCP for clustering and routing inside of the cluster [161]. VCP organizes the cars as a virtual cord with cluster positions ranging from 0.0 to 1.0. For routing purposes, messages are passed along this cord with the potential to exploit shortcuts. Furthermore, VCP supports cars entering and leaving the cluster. Finally, the virtual cord of VCP works like a DHT and allows CMs to store data, i.e., control information as well as data necessary for running services.

### 4.3.3 Gateway Selection

To connect to our virtual RSU, moving cars use messages they receive from *gateways*. If all cars are gateways, i.e., all of them periodically announce their presence via beaconing, and passing cars answer these beacons, multiple problems arise:

- Considering a parking lot, cars are parking close to each other. If all cars would send beacons to announce the clusters existence, many of those messages would be redundant for cars intending to connect.

- Furthermore, if all of the cars periodically send messages, too much load would be put onto the wireless channel.

- Cars surrounded by other cars do not need to be gateways as there are usually cars better suited to be connected to. This is especially true in a parking lot with multiple rows of cars.

To avoid the outlined issues, we propose an algorithm, which selects a subset of the parked cars as *gateways*. Beside these gateways, no other car should provide access to the micro cloud. Therefore, the set of selected gateways should fulfill two properties: (1) be at the perimeter of the cluster and (2) select only the absolute necessary number of cars to reduce the load.

After this informal description, in the following we outline the concept of gateways in a more formal way. The objective is to cover a region with a set of gateways $\mathbb{G}$ to reduce the regions covered by 2 or more gateways to a minimum. To solve this in an optimal way, all wireless coverage information from the parked cars is necessary. By assuming a fixed transmission distance, this can be solved easily. Nevertheless, this will only lead to a rough abstraction as in reality, such a unit-disk model is not sufficient. This is because coverage changes over time and space. Consequently, we approximate $\mathbb{G}$ based on information, which can be easily provided by the cars: the 1-hop neighborhoods of all cars and the geographic position.

One dedicated CM acts as the coordinator and calculates which cars are gateways. In case of VCP, we chose to select the node with cord position 0.0 as the coordinator. This node is always available due to how VCP works. To calculate the gateways, nodes send the necessary information to the coordinator. Note that, this is only necessary when something in the cluster changes. Based on this information, the selection algorithm calculates the gateways in two ways: by selecting the gateways along a curve (Algorithm 4.1) or by selecting them in an area (Algorithm 4.2).

The first algorithm selects the set of gateways $\mathbb{G}$ based on a set of nodes $\mathbb{N}$ along a curve. As the input to this algorithm is a set of nodes $\mathbb{N}$, not all cars need to be included. Any subset of selected cars can be used. This becomes important later, when we discuss how to calculate $\mathbb{G}$ for an area. Defined in Algorithm 4.1, the algorithm performs the following steps:

1. Select the node farthest away from all other cars and add it to the set of gateways $\mathbb{G}$.

2. Iterate over all remaining nodes and select the node farthest away from all nodes in $\mathbb{G}$. Do not immediately add the node to $\mathbb{G}$, but rather note it as a candidate node.

3. Count the number of 1-hop neighbors of the candidate node, which are already in $\mathbb{G}$. If more than 2 are already a gateway, all positions are covered by at least two gateways and the selection algorithm stops. Otherwise, the candidate node is added to $\mathbb{G}$ and the process of selecting the farthest node from all gateways is performed again.

An example of the selection process is depicted in Figure 4.4 with connectivity shown as simple unit-disk models. First, the cars A and B are selected as gateways.

**Require:** $\mathbb{N}$, a set of nodes
**Ensure:** $\mathbb{G}$, the set of nodes selected as gateways
1: $n \leftarrow \arg\max_{n \in \mathbb{N}} \sum_{n_x \in \mathbb{N}} \text{distance}(n_x, n)$
2: $\mathbb{G} \leftarrow \{n\}$
3: $\mathbb{N} \leftarrow \mathbb{N} \setminus \{n\}$
4: **while** $\mathbb{N} \neq \varnothing$ **do**
5: $n \leftarrow \arg\max_{n \in \mathbb{N}} \sum_{n_g \in \mathbb{G}} \text{distance}(n_g, n)$
6: **if** $|\text{neighbors}(n) \cap \mathbb{G}| \geq 2$ **then**
7: break
8: **end if**
9: $\mathbb{G} \leftarrow \mathbb{G} \cup \{n\}$
10: $\mathbb{N} \leftarrow \mathbb{N} \setminus \{n\}$
11: **end while**
12: **return** $\mathbb{G}$

**Algorithm 4.1** – Gateway selection along a curve.

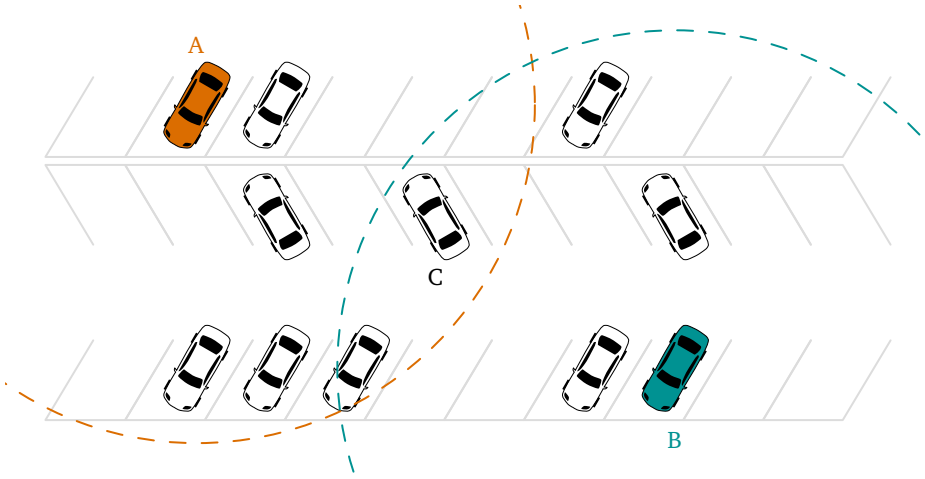


**Figure 4.4** – An illustration showing the halting criteria for the proposed gateway selection algorithms. As two neighbors of C, A and B, are already gateways, C will not become a gateway and the algorithm ends; based on [55] © Elsevier B.V.

The next candidate would be car C, but it is not added to $\mathbb{G}$ as two of its neighbors, A and B, are already part of $\mathbb{G}$. Moreover, the selection process is finished.

The second algorithm selects the set of gateways $\mathbb{G}$ based on a set of nodes $\mathbb{N}$ in an area. We require such an algorithm as the selection along a curve does not work well if cars are parked in a parking lot. In such a scenario Algorithm 4.1 could produce a set $\mathbb{G}$ with unwanted gateways located within the cluster. Therefore, Algorithm 4.2 first selects cars on the perimeter of the parking lot resulting in a concave hull [162]. Calculating only the convex hull would not be enough as it easily

**Require:** $\mathbb{N}$, the set of all nodes in the cluster
**Require:** $\Delta$, digging parameter
**Ensure:** $\mathbb{G}$, the set of nodes selected as gateways
1: $\mathbb{C} \leftarrow$ edges of the convex hull of $\mathbb{N}$
2: **for all** $c \in \mathbb{C}$ **do**
3: $c_0 \leftarrow$ start of $c$
4: $c_1 \leftarrow$ end of $c$
5: find $p \in \mathbb{N} \setminus \{\text{points on } \mathbb{C}\}$ closest to edge $c$
6: $\varsigma = \frac{\text{distance}(c_0, c_1)}{\min\{\text{distance}(c_0, p), \text{distance}(p, c_1)\}}$
7: **if** $\varsigma > \Delta$ **then**
8: remove $c$ from $\mathbb{C}$
9: add edges $(c_0, p)$ and $(p, c_1)$ to $\mathbb{C}$
10: **end if**
11: **end for**
12: $\mathbb{G} \leftarrow$ Algorithm 4.1 with {points on $\mathbb{C}$} as input
13: **return** $\mathbb{G}$

**Algorithm 4.2** – Gateway selection in an area.

leads to areas without sufficient coverage. The digging parameter $\Delta$ allows tuning which cars from the perimeter will be selected, i.e., how deep the algorithm digs. After calculating the concave hull, locations of the selected cars can be interpreted to be along a curve. This is ideal as we are now able to use the concave hull as input for Algorithm 4.1 to finally select the gateways $\mathbb{G}$.

The concrete steps performed by Algorithm 4.2 are as follows:

1. Calculate all edges of the convex hull and store them in set $\mathbb{C}$.

2. For every edge $c \in \mathbb{C}$, find the point $p$ closest to it. Calculate the distance factor $\varsigma$, i.e., the length of $c$ divided by the minimal distance between $p$ and either the start or end of $c$. If $\varsigma$ is smaller than the digging parameter $\Delta$, the edges from start and end of $c$ to $p$ are added to $\mathbb{C}$ and $c$ is removed from $\mathbb{C}$.

3. The resulting edge set $\mathbb{C}$ is used as an input to Algorithm 4.1, which yields the set of gateways $\mathbb{G}$

In Figure 4.5 we show an example of both algorithms selecting gateways. First, we rely on Algorithm 4.2 to select the concave hull. To get the concave hull, Algorithm 4.2 starts with selecting the convex hull (dotted line). Building upon this, additional cars are added to create the concave hull (solid line) and, therefore, getting a longer perimeter with less potential coverage holes. Afterwards, Algorithm 4.1 operates based on the selected cars. As outlined, we first select the car the farthest away from all others and add it to the set of gateways $\mathbb{G}$ — in this example the car is A. Now, in a loop, other cars are added based on which one is the farthest away from all

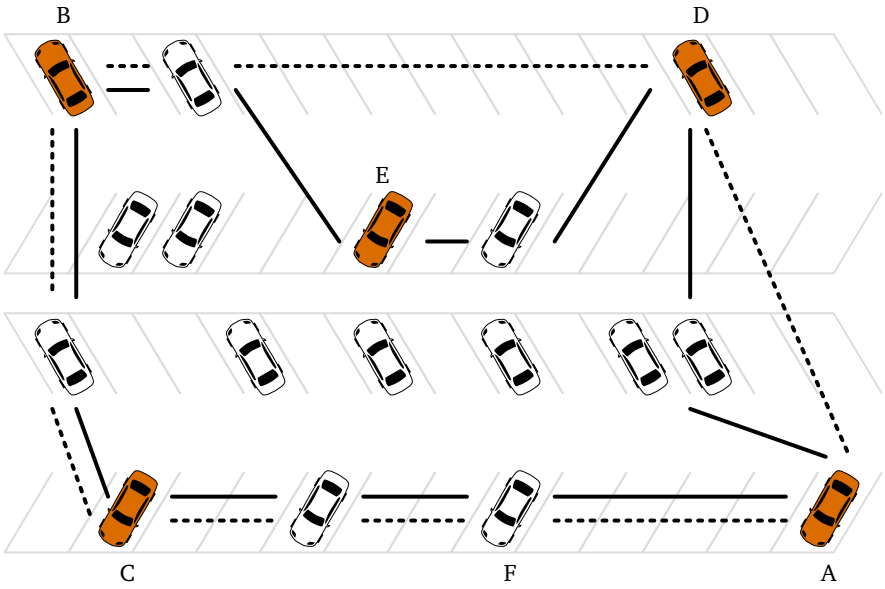


**Figure 4.5** – Overview of the gateway selection algorithm used in parking lot, i.e., in an area. Initially, the convex hull is created (dashed line). This is further extended to form the concave hull (solid line). Afterwards, the gateway candidates are used as input to Algorithm 4.1 as they now form a curve. The final selected gateways are highlighted; based on [55] © Elsevier B.V.

gateways. In our example, the cars would be in order B, C, D, and E. Finally, car F is selected as a candidate. But, due to it having two neighbors which are already part of $\mathbb{G}$, A and C, the algorithm terminates and yields $\mathbb{G} = \{A, B, C, D, E\}$. The resulting gateways are marked with wireless signals in our example. Overall, in this example roughly 30 % of cars are gateways.

### 4.3.4 Handover

A big advantage of having such a large micro cloud is the potential longer contact times between a passing car the micro cloud. This enables more complex protocols and the transfer of large files. To provide this functionality, below we outline our approach to near-seamless handover. This enables cars to connect to multiple gateways while passing by the cluster with minimal overhead. If a car is connected to the cluster, there are two ways how data is exchanged: (1) receiving data from the cluster, i.e., *downloading* and (2) sending data to the cluster, i.e., *uploading*.

To enable the proposed handovers, cars intending to exchange data with the cluster need to reply to *cluster beacons*. These replies contain the information necessary to send data to the car:

- **Receiving Gateway Identifier**: This indicates which gateway received the beacon and is unique among all gateways. For example, when using VCP, this is the cord position.
- **Destination Identifier**: A unique identifier for the car downloading data. This should be unique in regards to the cluster, but can be altered when communicating with other clusters.
- **Contact Time**: The time when the gateway received the reply to the beacon.
- **Distance**: The distance between the passing car and the gateway.
- **Signal to Interference and Noise Ratio**: The SINR is used as a metric to measure the connectivity.

For downloading data, the cluster stores the received information in the DHT and uses it to determine the best gateway for sending data to the passing car. More formally, we want to send data to a passing car $c$ and decide which gateway $g' \in \mathbb{G}$ is suited best. For this, a weight $w$ is calculated for all potential gateways, i.e., the gateways which received a reply from $c$. The weight for gateway $i$ is calculated as

$$w(i) = \alpha \times d_i + \beta \times t_i - \gamma \times s_i \ , \tag{4.1}$$

with parameters being $d_i$, the distance in meters, $t_i$, the time passed since the last received reply in seconds, and $s_i$, the SINR of the last reply. Furthermore, we have three coefficients $\alpha$, $\beta$, and $\gamma$ representing the weights for the three parameters ($\alpha + \beta + \gamma = 1$). The best gateway $g$ is calculated as:

$$g' = \min_{g \in \mathbb{G}} w(g). \tag{4.2}$$

Weights are required as the value range of the parameters is vastly different. For example, while the distance can be estimated to be between 0 m and 500 m, the last contact time only ranges between 0 s and 5 s. After calculating $w$ for all potential gateways, the gateway with the lowest $w$ becomes the destined $g$. For every sent packet, the sending CM must perform this calculation again. As the gateways themselves do not need to calculate anything, this can be considered a passive handover.

Uploading data from a passing car $c$ to any CM of the micro cloud also exploits the gateway beacons. Upon receiving such a beacon, $c$ stores the provided access information. Whenever $c$ intends to upload data, it can choose any known gateway to upload the data to. The gateway will be able to determine the correct receiving CM and sends the data to it (in case of using VCP, this is done by routing along the cord). This works for all gateways as we require all CMs to be interconnected.

### 4.3.5 Evaluation

The evaluation of our proposed virtual network infrastructure is split into two parts. First, we investigate the impact of gateway selection and handover on the performance. Second, we evaluate how the complete architecture performs in two different real-world scenarios. This helps us in understanding the benefits of using the architecture as well as the limits it imposes. Our evaluation was done using simulation, namely Veins [98] for simulating network and road traffic and its Veins LTE extension [49].

There are many possible metrics when evaluating the proposed architecture. We used the following as we think they allow to grasp the performance of the system along different angles:

- **Success Rate**: This metric covers the fraction of sent frames which have been received by the destination. Depending on the scenario, the sender can be a moving car and the receiver the cluster or vice versa. The metric provides insights into multiple characteristics of the micro cloud architecture. Moreover, with it we can investigate how well the transfer of a large file or continuously streaming data.

- **MAC busy-fraction**: This metric covers the fraction of time the MAC layer considers the channel busy. We use it to assess the congestion of the wireless channel. It not only enables us to confirm performance drops but also indicates if there is any capacity left for other applications — which is preferable.

- **Handover Count**: This metric covers how many gateways a car connects to while communicating with the micro cloud. If the number of connected gateways is too large, this might indicate a sub-par selection algorithm.

To assess the performance of our algorithms, we used three scenarios: one artificial and two real-world scenarios. While the artificial scenario was handcrafted, we extracted the real-world scenarios from the Luxembourg SUMO Traffic scenario [97].

The artificial scenario contains 50 cars parked along a straight road with 30 m between them. Such a basic scenario allowed us to investigate the effects of gateway selection and handover in a controlled environment. The realistic scenarios are based on two different areas of Luxembourg, each with different surroundings and geometries. First, one scenario covers a junction close to the train station (Figure 4.6). Beside a busy junction, there are two parking lots close to each other in the scenario. They are considered as a single micro cloud. One CM of the micro cloud provides data, which is requested and downloaded by passing cars. Second, the other scenario covers a large parking lot along a freeway leading into the city (Figure 4.7). The parking lot works as a *park-and-ride* where commuters leave their cars and continue

**Table 4.1** – Simulation Parameters for the parked car simulations.

| Parameter | Value |
|---|---|
| IVC technology | IEEE 802.11p |
| Channel | 5.89 GHz |
| Transmission power | 20 mW |
| Bandwidth | 10 MHz |
| DHT protocol | Virtual Cord Protocol |
| Routing | greedy VCP routing |
| Gateway digging parameter $\Delta$ | 1 |
| Artificial Scenario | |
| Parked cars | 50 |
| Driving cars | on average 1 every 10 s |
| Amount of requested data | 128 kB |
| Simulation duration | 360 s of traffic |
| Repetitions | 32 |
| Real-world Scenario 1 | |
| Parked cars | 11 west, 9 east |
| Car4ICT service provider | one at west parking lot |
| Fraction of equipped vehicles | 0.25, 0.5, 0.75, 0.1 |
| Simulated time | 600 s during rush hour |
| Real-world Scenario 2 | |
| Parked cars | 40 cars |
| Car4ICT service provider | on at the parking lot |
| Fraction of equipped vehicles | 0.25, 0.5, 0.75, 0.1 |
| Simulated time | 600 s during rush hour |

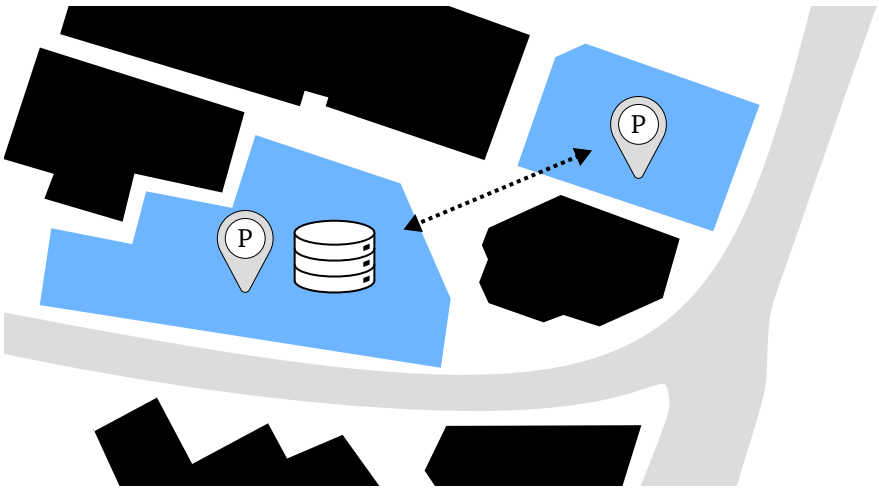


**Figure 4.6** – Scenario 1: Two parking lots along a busy intersection close to Luxembourg main station; based on [55] © Elsevier B.V.

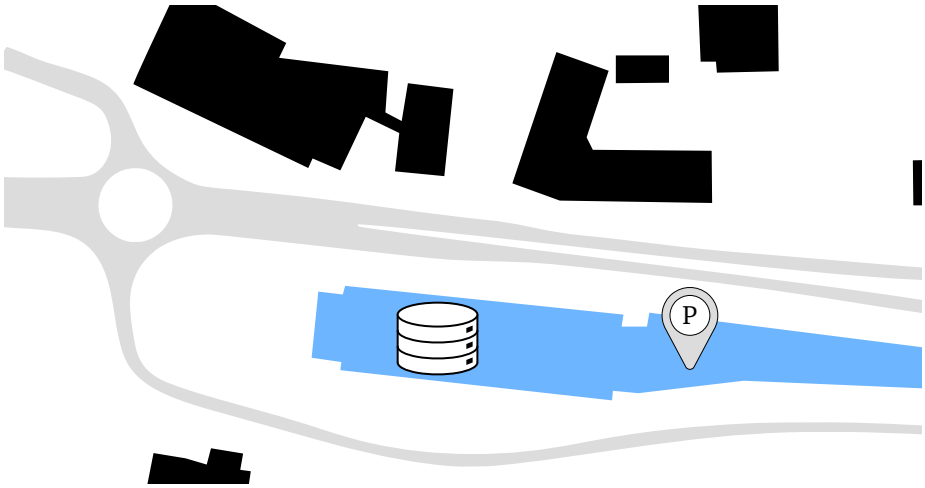


**Figure 4.7** – Scenario 2: Large parking lot along a highway; based on [55]


by public transportation. Due to the freeway, cars potentially pass with fast speeds along the cluster leading to a short contact time and fast handovers.

In addition to the micro cloud architecture we used the Car4ICT framework [50]. We used it to delegate service provider and consumer roles to parked and moving cars and how data is exchanged between them. The parameters we used in our simulation can be found in Table 4.1. Unless otherwise noted, we show the 95 % confidence interval in the plots of this section.

#### 4.3.5.1 Gateway Selection Performance

Using the artificial scenario, we started our evaluation regarding the gateway selection. By putting the focus on the MAC busy-fraction we gathered insights regarding the overhead of control messages. As a baseline we used the performance without gateway selection, which leads to all cars sending beacons and acting as gateways.

First, in Figure 4.8 we show the influence gateway selection has on the wireless channel load. The biggest difference can be observed for an access beacon interval of 0.5 s. On the one hand, if all cars are gateways, the channel is busy nearly 20 % of the time. On the other hand, if gateways are selected, the load is roughly 4 times lower, even below 5 %. This effect can be observed for the other intervals but the absolute difference becomes smaller. Based on this observation we can clearly see that gateway selection is able to reduce the load significantly even for relatively small beacon intervals.

In our algorithm, cars can take three different roles regarding gateways: being a gateway, being a parked car but no gateway, or being a passing car. The different effect of these roles on the channel load can be seen in Figure 4.9. The lowest load is

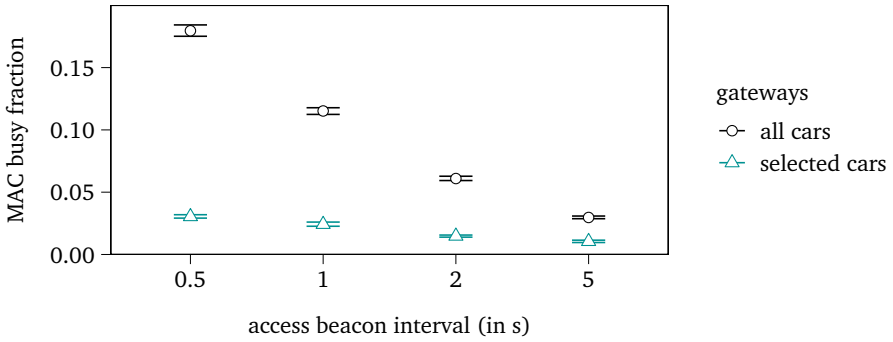


**Figure 4.8** – The fraction of time the MAC is busy in the artificial scenario. Parameters are access beacon intervals in the range of 0.5 s . . . 5 s and if gateway selection is performed or not; based on [55] © Elsevier B.V.

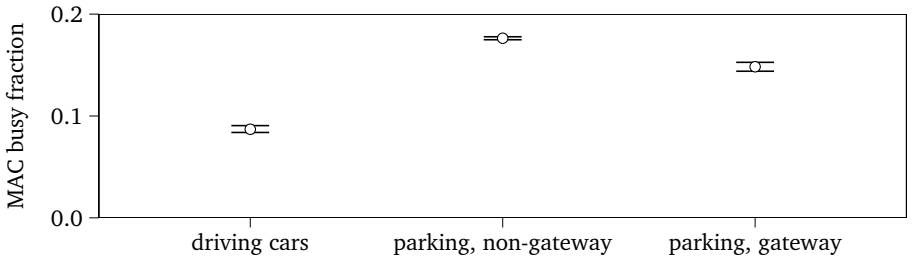


**Figure 4.9** – The fraction of time the MAC is busy for driving and parked cars. There is a second distinction for parked cars being selected as gateways and those which are not.; based on [55] © Elsevier B.V.

observed by the driving cars as they only observe load on the channel while passing the cluster and neither before nor after. Interestingly, CMs which are not gateways observe a higher load compared to actual gateways. This is a result from the gateway selection algorithm preferring cars on the fringe of the cluster. Therefore, cars inside the cluster receive more messages than gateways.

The more up-to-date the available control information is, the better are the handover results. To provide useful information while not increasing the load on the channel too much we set the access beacon interval to 1 s for the other simulations.

#### 4.3.5.2 Handover Performance

The handover performance can be best evaluated if many handovers are happening. Therefore, we configured the artificial scenario so that the car offering the data sends 1 kB every 0.05 s instead of larger fragments. Data is transferred using a continuous streaming process in which the driving car needs to perform multiple handovers

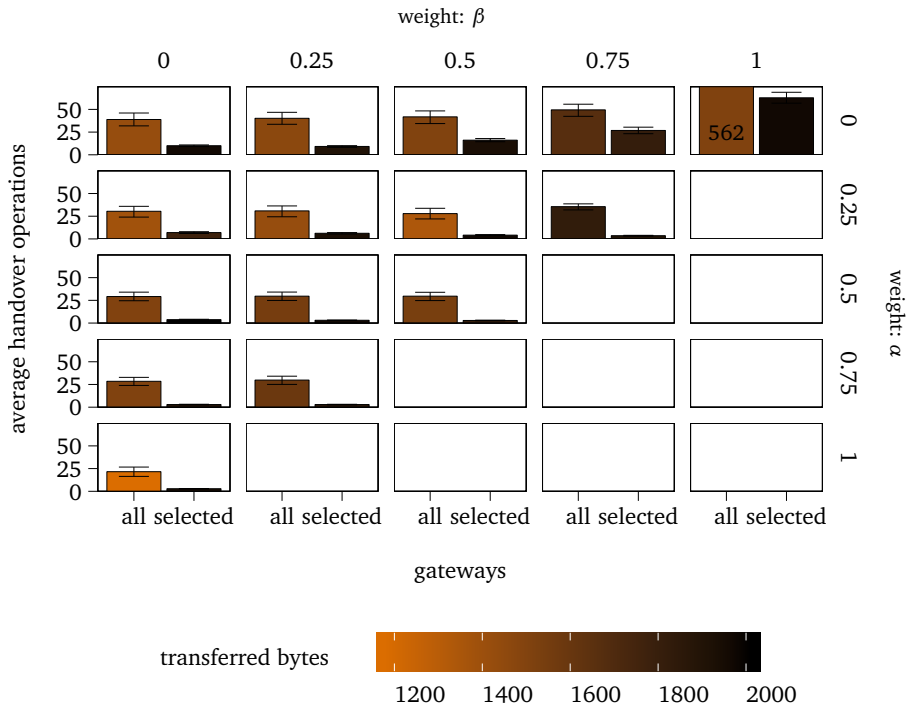


**Figure 4.10** – Handover counts for different values of $\alpha$, $\beta$, and implicitly, $\gamma$. Furthermore, the shade shows how many streamed bytes have been received; based on [55] © Elsevier B.V.

while passing the cluster. Like our gateway selection evaluation, we also investigate the difference between gateway selection and all cars being gateways.

As metrics, we have chosen to use the number of handovers and the amount of received data. If a car needs to perform numerous handovers, this indicates that unnecessary handovers are performed, which points towards a poor selection of gateways. The other metric, the amount of received data, is an indicator how well the overall system works. Both metrics can be seen in Figure 4.10 where they are shown for different weights of $\alpha$, $\beta$, and, implicitly, $\gamma$ (as $\alpha + \beta + \gamma = 1$).

Based on the shown results, there are interesting observations to be made:

- First, if the distance is ignored when calculating the weight (i.e., $\alpha = 1$) the number of handovers is higher compared to other configurations. This can be an indication that incorporating SINR improves the performance in cases where distance leads to wrong results (e.g., a car close to a gateway but with poor connection to it). Otherwise, the top-most line would be like the diagonal (except for $\beta = 1$ and $\alpha = 1$). This is because the diagonal line does not incorporate SINR and the top-most line does not incorporate distance. If they would cover the same scenarios, these rows would be nearly the same.

- Second, if handover is performed only based on time (i.e., $\beta = 1$), the handover count is the highest, especially if no gateways are selected. This behavior is due to how the handover algorithm selects which gateway to send to. If more than 1 gateway is in reach of a passing car, it periodically switches between the two, whichever it replied last to. Nevertheless, this regular switching does not impair the performance of downloading data. This is because, for every data fragment, the best gateway is selected, even if it means switching for every single one.

- Third, if all cars act as gateways, the handover count is larger compared to if only selected cars are gateways. Due to the higher number of potential gateways and no filtering, the chance of connecting to an imperfect gateway increases. If this happens, packets get lost and less data is successfully transferred — this effect can be seen in all configurations, but is weaker when the time is given more weight, i.e., having a high $\beta$.

When only looking at the performance, i.e., how many bytes have been successfully transferred, we can see that multiple weight combinations work well. This also includes the ones only relying on a single parameter. Some parameter combinations enable a poor gateway selection, e.g., when relying too much on $\beta$, the gateway the farthest away might be selected as it received the last message. If we exclude those combinations we are left with two parameter combinations: ($\alpha = 0.5$, $\beta = 0$, $\gamma = 0.5$) and ($\alpha = 0.25$, $\beta = 0.5$, $\gamma = 0.25$). These two lead to good results regarding both metrics: the number of handovers is small and the number of received streamed data is high.

The other way of sending data would be not to wait in-between sending fragments, but send them all together. We investigated the effects of this in Figure 4.11 where the provider sends 128 kB immediately. In addition, a steady stream of cars passes the

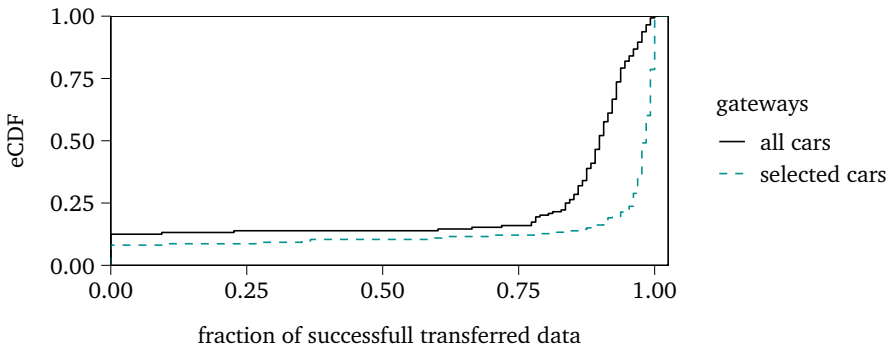


**Figure 4.11** – Results showing the effect of handover onto the amount of transferred data; based on [55] © Elsevier B.V.

micro cloud instead of just a single car. By doing this, we evaluated how successful the combination of gateway selection and handovers is compared to our baseline. As the results indicate, by enabling gateway selection in this scenario, we lose some performance. While the fraction of received data is lower, we still argue that the trade-off is acceptable in order to significantly reduce the load on the channel as the previous results indicated.

#### 4.3.5.3 Real-World System Performance

So far, we investigated the performance of our micro cloud architecture in an artificial scenario. The results indicated good performance and allowed us to start with looking into real-world settings. For this, we created two scenarios, shown in Figure 4.6 and Figure 4.7. In those we were running two applications:

- Passing cars download 512 kB from a provider located inside the cluster. When the provider receives a request it immediately sends all data towards the passing car, therefore, choosing only a single gateway.
- Passing cars download a constant stream of 0.25 kB fragments every 0.2 s from the provider. This is only started after a request was received by the provider.

For all scenarios we also used the penetration rate of driving cars being equipped with communication technologies as an additional parameter. This allowed us to acquire insights how such a system would work during its roll-out phase.

Figure 4.12 shows the success of the direct download of 512 kB. The more cars intending to download the data, the lower the average success of downloading it. In the first scenario, the intersection close to the main station, the effect is much stronger. In the second scenario there is only a drop between 25 % and 50 % due to receiving more erroneous frames. We are not able to observe the same increase of errors

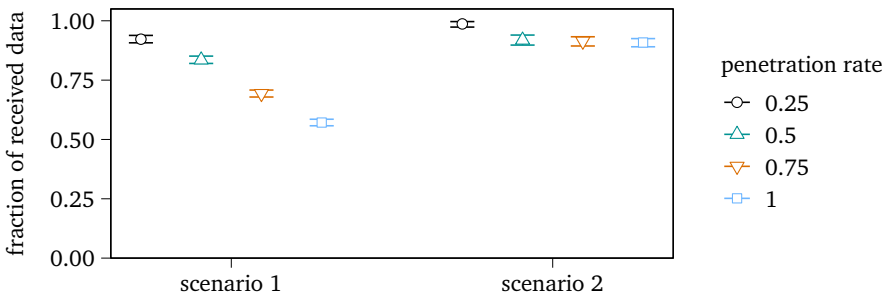


**Figure 4.12** – Results showing how much of the data (up to 512 kB) have been successfully transfered. Parameters are the different penetration rates and the simulations have been done for both scenarios; based on [55] © Elsevier B.V.

for the other two cases. Note that there is no significant difference when disabling handover completely in this experiment, as the download is anyway handled by a single gateway.

The reason for the decreasing performance with increasing penetration rate is because of the channel load. This can be seen more clearly in Figure 4.13, where we show the MAC busy-fraction for the different configurations. By looking at the results for scenario 1 we can see that indeed the load gets too high to allow good performance. For a penetration rate of 50 % the load is on average below 50 % but for higher penetration rates it increases above 60 %. As the traffic near the main station is dense, there are many cars requesting data and downloading it from the micro cloud. This leads to a high channel load, especially for a high penetration rate.

When downloading 512 kB, having handover enabled or not, did not make a large difference as most of the data is sent via a single gateway. But, when we

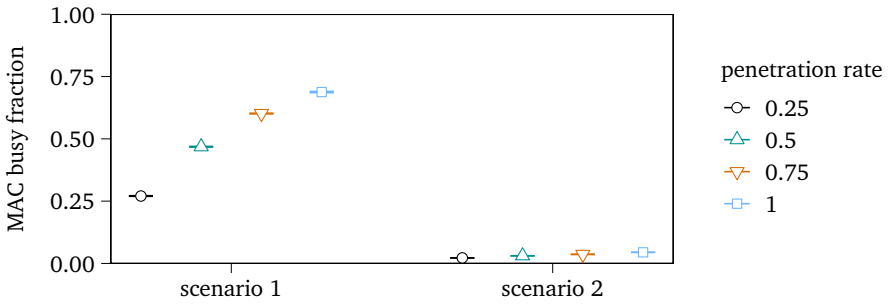


**Figure 4.13** – Results showing how much the MAC was observed busy while transmitting 512 kB of data. Parameters are the different penetration rates and the simulations have been done for both scenarios; based on [55] © Elsevier B.V.

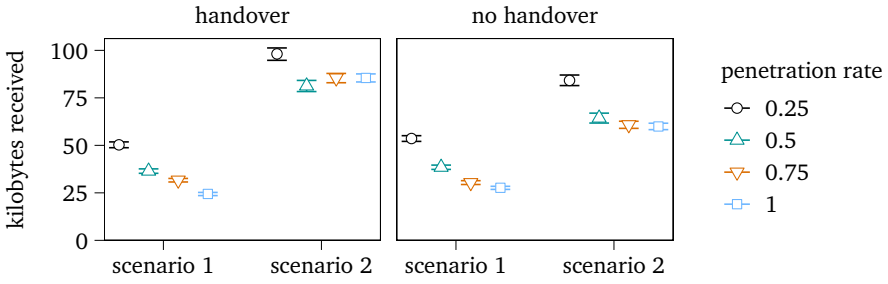


**Figure 4.14** – Results showing how many kilobytes have been received when data is streamed, i.e., 0.25 kB every 2 s; based on [55] © Elsevier B.V.

consider streaming, having handover makes a difference, especially in the second scenario. We show the results in Figure 4.14. When comparing the results for the second scenario regarding handover we can see that handover improves the performance by 15 %...30 %.

We can make more observations showing the difference between our scenarios:

- Cars requesting data in the first scenario receive much less data overall compared to the second scenario. This is a result of the micro cloud length along the street. While in the first scenario, the cloud covers a length of roughly 150 m, the second scenario covers approximately 500 m. Due to this, cars can download more data in the second scenario because the contact time is longer.

- Handover can improve the systems performance. Especially in scenario two, where cars drive along the cluster for a longer time, handover enables longer connections between the micro cloud and the passing car. However, due to being smaller and having only a shorter contact time, enabling handover makes barely any difference in the first scenario. This is because cars usually are only connected to 1 or 2 gateways.

The last set of results is shown in Figure 4.15 where we investigated the MAC busy-fraction in our realistic scenarios. The data shows yet another reason the performance in the first scenario is worse. Compared to the second scenario, the load on the wireless channel is much higher. This leads to more erroneous and delayed transmissions reducing the performance of the system. Therefore, there is room for further improving the algorithm by using load balancing in dense scenarios.

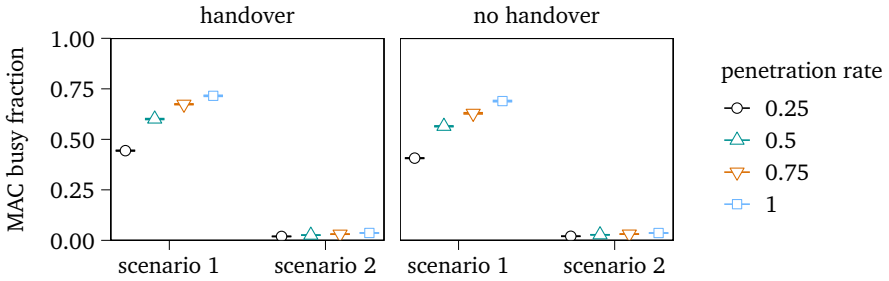


**Figure 4.15** – Results showing how often the MAC was observed busy when data is streamed, i.e., 0.25 kB every 2 s; based on [55] © Elsevier B.V.

### 4.3.6 Lessons Learned

In this section we outlined a concept for forming micro clouds using parked cars. Driving cars connect to these clouds and exchange data, essentially turning the micro clouds into virtual network infrastructure. To reduce the load on the wireless channel, certain cars of the cloud are selected as gateways and only these can be used to initiate connections. While passing, a car connects to multiple gateways while the connection is handed over between them enabling the transfer of larger files or using more complex protocols.

We were able to show that connecting only to these gateways reduces the load on the wireless channel while barely affecting the amount of successfully transferred data. Nevertheless, there is still room for improvement to make the gateway selection on-par to having none. Furthermore, we were able to demonstrate that handing over connections allows cars to download more data from a cluster. This is especially true for larger clusters where a car passes multiple gateways and continuously streams data. Still, channel load can become an issue and reduces the amount of received data. This is backed by literature and is considered a main issue leading to the development of adaptive protocols [27], [75]. Here, further research might be needed in the context of micro clouds and clustering. One idea could be to use an even more passive clustering concept, which is able to keep functioning with a low number of messages. Furthermore, adaptive concepts and load balancing are a direction with the potential to improve the performance when transferring data.

## 4.4 Micro Clouds with Moving Cars

Micro clouds cannot only be formed using parked cars, but also with moving cars. In this section we present an approach for stationary micro clouds using driving cars. Furthermore, compared to the approach outlined in the previous section, the micro cloud coordination is not done in a distributed manner, but by a centralized coordinator. Any Access Point could take such a coordination role, especially if they are placed at beneficial locations throughout a city. Such a location could be an intersection enabling communication in many directions. The resulting micro clouds are formed using clustering concepts where one car is a CH and others take the role of CMs. Beside outlining this generic concept, we present two services running in such a micro cloud. One takes care of collecting data and uses the CH to upload it to the macro cloud for further processing. The other service preserves locally relevant data in the micro cloud essentially offloading the macro cloud by using local processing and storage resources. Both these services can be seen as a foundation for user-facing applications, e.g., TISs, or offloading using MEC concepts.

### 4.4.1 Micro Cloud Architecture

Figure 4.16 outlines the purpose of our proposed micro cloud architecture and how it fits into the MMC concept (micro cloud C is an example in the Figure). Several micro clouds form a macro cloud, which is then connected to a back-end data center. Altogether, a vast number of potential services can be provided with micro clouds at the center. In this section, we discuss two such services — one for collecting information about the micro cloud and one that preserves data locally to offload the macro cloud and the data center.

The *collect* service enables CHs to gather data from their CMs and upload it to the macro cloud. CM and especially CHs can pre-process the data before uploading it by employing aggregation techniques. This relieves the macro cloud and the data center from unnecessary data processing and might also reduce the load on the channel. Collected data could be Floating Car Data (FCD) and consequently has various use-cases. For example, by collecting historical data and combining it with live data it can be possible to improve the traffic flow in a city. Another possibility would be to provide live route information for a driver based on recent information from micro clouds along the route.

The *preserve* service provides completely different functionality as it does not upload the data, but rather keeps it locally available at the micro cloud. Data fragments can be stored by CMs and are replicated to improve availability. Using this, locally relevant information is cached in the micro cloud, essentially offloading

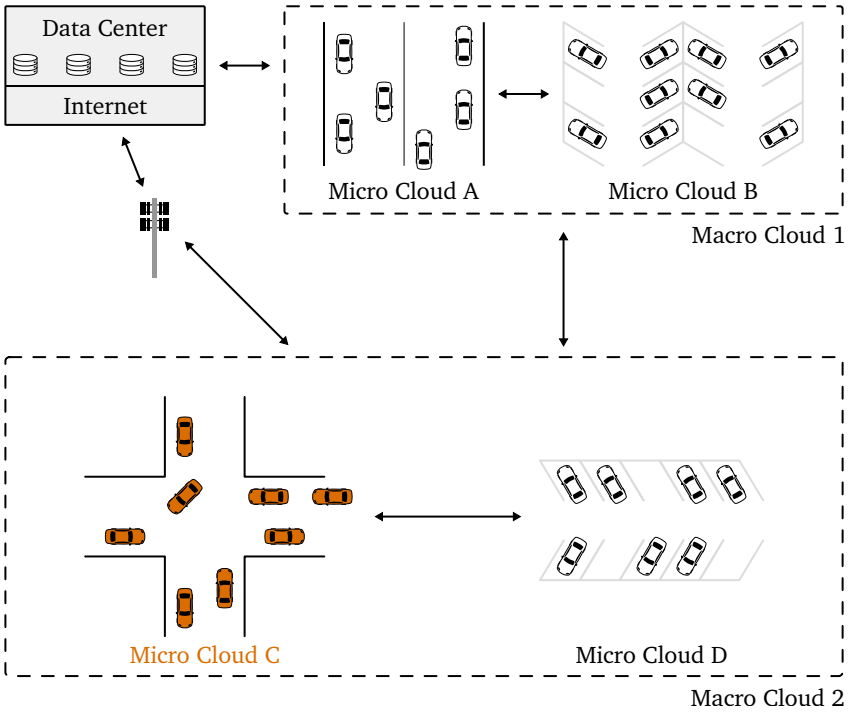


**Figure 4.16** – Overview of the MMC architecture from Chapter 2 where the micro cloud (A) considered in this chapter are highlighted.

data from the macro cloud and/or the data center. This works well as many of the envisioned services rely mostly on such local data.

The enabling architecture for these services is *geographic clustering*, which creates clusters at beneficial locations to improve in-cluster connectivity [56], [57]. Such clusters consist of driving cars and are located mostly around junctions. The car closest to the center of the cluster is selected as CH and other cars are placed into the cluster to which CH they are closest to. By doing this, we intend to optimize the in-cluster communication especially between the CH and its CMs.

### 4.4.2 Requirements

There are certain requirements to make our proposed architecture work. Basically, all these requirements are fulfilled by the connected cars envisioned to be ubiquitous in a smart city:

- **Vehicle-to-Vehicle Communication**: To communicate with cars in the vicinity, appropriate communication technology should be available. We do not limit the choice and it is possible to use any Device-to-Device (D2D) technology, for instance Dedicated Short-Range Communication (DSRC), WLAN, or LTE-D2D.

- **Vehicle-to-Infrastructure Communication**: To communicate with APs, cars must be equipped with suitable technology. There is no reason the V2V technology cannot be used for this as well, but it can also be done by relying on something entirely different. Potential examples include LTE or communicating on a separate wireless channel.

- **Geographic Positioning**: To distribute the cars into clusters, it is mandatory to know their current geographic location. Usually, this information is provided by the cars themselves using a GNSS. Examples of such systems are GPS, GLONASS, or Galileo.

- **Macro Cloud Connection**: To unlock the full potential of the proposed MMC system, the micro cloud needs to have a connection to the macro cloud. Such a connection is necessary for various services, e.g., to upload data when using the *collect* service, or to request locally stored data for the *preserve* service. Generally, any car with a cellular connection could be able to provide this. Nevertheless, in our architecture we exploit the AP for this and assume that they are connected to the macro cloud.

- **Selection of Cluster Locations**: To make our proposed concept work, cluster should be formed at suitable locations throughout a city. This information is necessary to deploy APs in their proximity. Mostly, we recommend placing micro clouds at intersections to enhance the in-cluster connectivity, but, depending on the employed services, a different factor might be deciding. A more detailed discussion of potential micro cloud placements is outlined by Higuchi, Dressler, and Altintas [45].

### 4.4.3 Geographic Clustering

Cooper et al. [44] outline the basic framework for forming and maintaining a cluster. For our proposed geographic clustering algorithm, we follow the frameworks steps while adapting and extending them where necessary. Overall, there are four steps necessary to go from cars with communication technology to full-fledged clusters:

1. **Gather Control Data**: To acquire the information necessary for calculating the cluster configuration, every car needs to periodically send a beacon. This beacon contains the required control information and is sent via broadcast. Therefore, all APs in the surroundings receive the information needed for putting the car into the best suited cluster. For clustering purposes, the most important control information is the cars current position. Consequently, the Cooperative Awareness Messages (CAMs) messages as standardized by ETSI ITS-G5 are sufficient [68].

**Require:** $\mathbb{J}$, set of all junctions.
**Require:** $\mathbb{C}$, set of all cars within communication range of the AP.
**Require:** $\mathbb{D}$, set $\{ d_{jc} \mid$ distance from junction $j$ to car $c \}$.
1: **for** $j \in \mathbb{J}$ **do**
2: Create a list of cars ordered by distance to the junction $j$.
3: First car in the list is a candidate CH of the cluster at $j$.
4: **end for**
5: **while** Two junctions have the same candidate CH $c$ **do**
6: Remove $c$ from the junction with the larger distance.
7: **end while**
8: $\mathbb{CH}$ = empty map of all CHs as keys, and a list of cluster members as value.
9: **for** $j \in \mathbb{J}$ **do**
10: $h$ = CH candidate for $j$.
11: Add a new cluster to $\mathbb{CH}$ with $h$ as a key and cluster head.
12: Remove $h$ from $\mathbb{C}$.
13: **end for**
14: **for** $c \in \mathbb{C}$ **do**
15: Add $c$ as a CM of the cluster at the junction $j$ whose distance $d_{jc}$ is the smallest.
16: **end for**

**Algorithm 4.3** – Selection of CHs

2. **Cluster Creation**: Periodically, the AP calculates the current cluster configuration based on received control information. This configuration is valid for the next interval, i.e., until the next calculation. Alongside distributing cars into clusters, cars are also assigned their cluster roles (CH or CM). This allows various services to exploit an additional hierarchical structure. The concrete algorithm can be found in Algorithm 4.3. To make it work with arbitrary configurations it does not rely on a 1-to-1 relationship of junctions and APs, but rather is able to calculate an arbitrary number of clusters. In the real-world, this means that a single AP manages multiple intersections, e.g., an eNB covering a larger part of a city. It takes two inputs: $\mathbb{J}$, the set of all junctions covered by this AP, $\mathbb{C}$, the set of cars the AP has valid control information. Based on $\mathbb{J}$ and $\mathbb{C}$, $\mathbb{D}$ is created, a set of all distances between cars and junctions. First, for all intersections the car closest to the intersection is selected as a candidate CH. Now, it might happen, that a car is candidate for two different intersections. Therefore, for every conflict, we remove the candidate from the junction which it has the largest distance to. Afterwards, the second-closest car becomes the new candidate. This process can lead to not all clusters having a CH. But, this is only the case, if there are more clusters configured than cars. Second, all candidate CH, which are now unique, are marked as CH. Third, for all cars not being a CH, put them as CM into the cluster which junction they are closest to.

3. **Distribute Control Information**: Every cluster configuration consists at least of a CH and can contain additional CMs. This information is distributed via broadcast by the AP to all cars in communication range. Whenever a car receives such information, it updates their current cluster role.

4. **Steady State**: As a final step, Cooper et al. [44] describe the collection of data. Our core clustering algorithm does not include such a step. Nevertheless, as this covers most of the time between the calculation of cluster configurations, this is the main step for any service to run.

Figure 4.17 shows two micro clouds A and B created using our geographic clustering algorithm. The cluster locations are the center of the two junctions. Based on this, the car closest to each junction became CH. Afterwards, the remaining cars were added as CM to the junction they are closes to. Furthermore, several cars are out of range of the APs and were not part of the current cluster configuration.

### 4.4.4 Micro Cloud Services

The outlined geographic clustering algorithm does not yet provide any value to the MMC architecture. But, the clustering enables services which can generate value to users. As core examples we propose and evaluate two fundamental services making such a micro cloud useful.

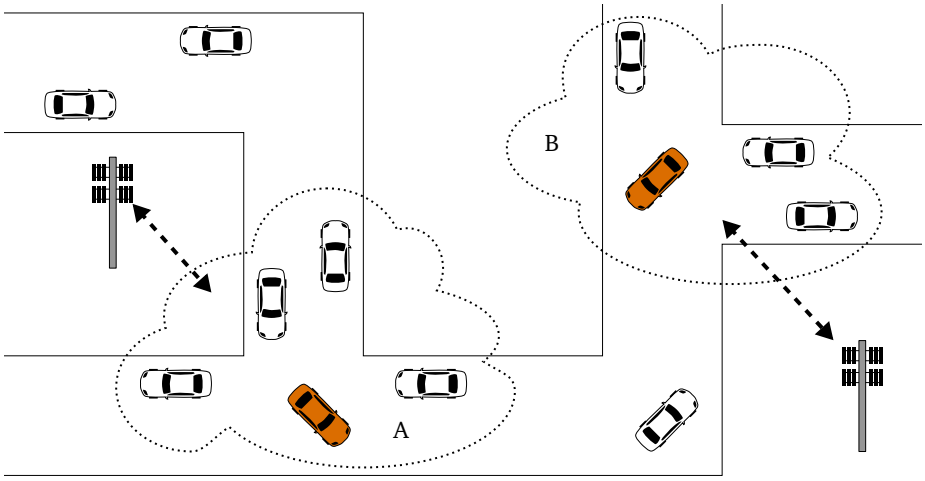


**Figure 4.17** – Two micro clouds, A and B, located a intersections. Both clouds have their own AP, which coordinates the cluster. Highlighted are the CHs, i.e., the cars closest to the center intersection; based on [57]

#### 4.4.4.1 Collect Data Service

The *collect* service acquires data from all CMs and uploads it to the macro cloud and in consequence to the data center. Such data usually consists of sensor data needed to provide services to users. For example, the data collected can be FCD. On the one hand, such data can be of use for the city government as it can be used to optimize traffic. On the other hand, such data can be used in a service providing traffic information for users before they start their journey.

The following steps are part of the *collect* service:

- Whenever a car receives or polls data from a sensor, it saves the data in a local KB. Potential sensor information includes geographic position, weather sensors like temperature, or multimedia data like images or videos.
- At a fixed interval, all CMs send data collected in their local KB to their CH.
- Every CH stores received data from its CMs. Before storing it, the CH might be able to pre-process the data, e.g., by aggregating similar information. The step of aggregating cannot be generalized as application dependent aggregation is able to provide by far the best performance [163].
- Periodically, the CH sends the collected and aggregated information to the AP, which in turn forwards it via the macro cloud to the data center.

When we evaluated the first version of our geographic clustering in combination with the *collect* service, we discovered that the link between CH and AP is the most critical [56]. If a transmission on this connection fails, the complete data from a collection interval is lost. Consider a cluster with $n$ CMs, each of them generating $b$ bytes of data. If the connection between a CM and the CH fails, $b$ bytes are lost. But, if the connection between CH and AP fails, independent of the used aggregation scheme, $(n+1) \times b$ bytes are lost. Therefore, it is necessary to rely on improvements focusing on this link, e.g., by employing Automatic Repeat Request (ARQ).

As already discussed, we want to be able to have an AP coordinating multiple micro clouds. If now multiple CHs try to simultaneously upload data, packet collisions, like the hidden terminal problem, might occur. By introducing a slotted upload system, we reduce such issues to a bare minimum. First, every CH waits for some time and collects data from its CMs. Afterwards, until the end of the cluster configuration interval, all CHs periodically upload data in their own upload slot.

To acquire the upload timings, we first split the cluster configuration interval $c$ into slots of length $u_g = c/(|u_g|+1)$. The initial slot is used for collecting data from the CMs, the later ones are used for collecting data and uploading it. To reduce the size of an upload and facilitate retransmissions we set $|u_g| = 2$. Besides having these slots, we provide individual upload slots $u_i$ for every CH. Uploads are started at the

beginning of $(n \times u_g) + u_i$, where $n$ is the current interval $u_g$. This is done to reduce potential collisions by parallel uploads of multiple CHs. Each of these slots has a length of $u_i = u_g/|\text{CH}|$, derived from the number of CHs managed by this AP.

Upon receiving cluster configuration information from an AP, a car verifies if it was selected as a CH. Afterwards, it waits for one upload slot $u_g$ before uploading during its individual slot.

To illustrate the process, we provide an example in Figure 4.18. We assume that the cluster configuration interval is $c = 12\,\text{s}$. As already discussed, two uploads should happen during this time, i.e., $u_g = 2$. Furthermore, for the example we assume that the calculating AP coordinates two CH nodes. First, the upload slots are calculated with a length of $|u_g| = 4\,\text{s}$. Afterwards, individual upload slots are set as $u_i = 2\,\text{s}$. The first CH waits for $1 \times u_g + 0 \times u_i = 4\,\text{s}$ before uploading while the second CH starts the upload after $1 \times u_g + 1 \times u_i = 6\,\text{s}$. Finally, the CHs upload again after 8 s and 10 s respectively.

#### 4.4.4.2 Preserve Data Service

The second micro cloud service we introduce does not upload data into the macro cloud, but rather keeps it locally inside the micro cloud. As many applications for vehicular networks rely on local data, instead of global/remote data this offloads processing and storing resources from the macro cloud. Examples of such data includes FCD used for traffic control, images and videos, or other location relevant data. To keep data fragments available even after a CM left the micro cloud, we require replication of data among CMs. By doing this, the micro cloud essentially caches data and acts as a storage extension. If all cars in a micro cloud have a data fragment available, we define a 100 % *replication rate*.

As a requirement for the *preserve* data service all data fragments need to have a unique identifier and a Time to Live (TTL). Such an identifier can be generated by hashing the data fragments content (e.g., using the *MD5* checksum). The TTL is used to determine the validity of data fragments and purge outdated ones. We extend the

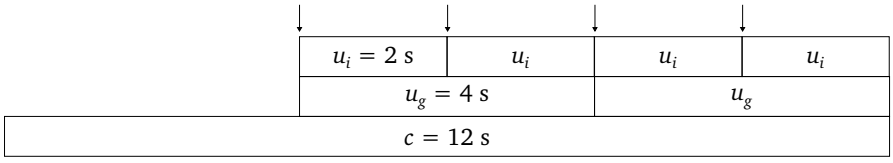


**Figure 4.18** – Exemplary calculation of the upload interval for each CH. Based on the cluster calculation interval $c$, four upload intervals ($u_i$) are calculated for the two CH. Arrows indicate when one of the CHs starts uploading their collected data; based on [57]

periodic updates sent to the AP with a list of all identifiers of data fragments stored in its KB. The AP compiles all these fragments into the *overall known data*. This is the data which should be available in the micro cloud, but due to lost fragments can be larger than the *actually known data*, describing which segments are currently known by cars. At fixed intervals, the AP uses set difference to determine missing data fragments for every car and sends this information via broadcast to all CMs. On receiving this information, every CM requests missing data from other CMs. To not overload the channel with parallel requests, all cars wait for a random interval before requesting missing data fragments. Fulfillment of a request is then done via unicast. Data fragments are generally purged from the KB on two occasions: (1) if a car leaves a cluster or (2) if the TTL of the fragment expires.

Table 4.2 shows an example were a cluster should have preserved 5 data fragments and which fragments the CMs are missing. While $v_1$ only misses two of the segments, $v_2$ seems to have recently joined as it misses all except for one segment. Note that, data fragment B is lost as no CM has stored it in its local KB.

By dividing the *actually known data* by the *overall known data* we can see how well the *preserve* service works. If the result is 1, all data is still preserved while lower values indicate missing data. Without any improvements such a data loss is permanent as data cannot be recovered. Furthermore, the load on the wireless channel is a good indicator of the performance. If the load is too high, there might be too many requests for missing data, which in turn lead to degrading performance of the *preserve* service. To improve the *actually known data* and potentially reduce the channel load we propose three improvements. These enable overhearing of requested data fragments, keeping data after leaving the micro cloud and a refinement how missing data is requested.

- **Overhear**: If a car requests a missing fragment, if it is available, it is sent via unicast. It can happen that multiple CMs request the same fragment leading to different cars requesting the same fragment. This is unfavorable as several transmissions increase the load on the wireless channel. If this happens, requested fragments might not be received, which in turn leads to even more

**Table 4.2** – An example of a cluster with 4 CMs and 5 data fragments.

| AP | $v_0$ | $v_1$ | $v_2$ | $v_3$ |
|---|---|---|---|---|
| A | ✓ | ✓ | | ✓ |
| B | | | | |
| C | ✓ | ✓ | | |
| D | | ✓ | | ✓ |
| E | | | ✓ | |
| | {B, D, E} | {B, E} | {A, B, C, D} | {B, C, E} |

missing fragments. The *overhear* improvement changes the transmission mode of requested fragments from unicast to multicast. Consequently, if more cars are missing the same fragment, only a single transmission is necessary. As cars wait a random interval before requesting missing data, the sender also aggregates requests for data fragments for some time before sending them. This is done to further reduce the number of transmissions.

- **Conserve**: Every time a car leaves a micro cloud, it purges all data fragments which originated from there from its KB. If a cluster consists of a small number of CMs, the chance of losing data, i.e., a low *actually known data*, grows larger. Such small clusters usually happen if the traffic density is low. The *conserve* improvement removes the requirement to purge data fragments of a left cluster — only outdated data fragments, i.e., with an expired TTL, need to be deleted. After joining a new cluster, a CM informs the AP of data fragments from other clusters. The AP then proceeds with these fragments in the same way as with others. Therefore, these fragments are now replicated as well. Later, a car might move the other way around and join the initial micro cloud later and in consequence return missing fragments. If the *overhear* improvement is activated as well, cars might be able to receive such fragments even if they currently are in another cluster.

- **Balanced Requests**: We also discovered potential for improvement in the way missing data fragments are requested. In the basic algorithm, for every missing fragment, a request is sent. Consequently, many unnecessary messages are sent, especially if a car is able to provide multiple missing data fragments. Our initial idea was to combine as many missing data fragments as possible into a single request. Unfortunately, this leads to many lost data fragments. This happened especially if a car possesses a large number of the fragments and tries to send them all. To prevent this, we added *balanced requests*: a car aims to request the same number of missing fragments from other CMs. Now, with this improvement, the process of requesting missing data leads to less lost fragments. Moreover, balancing the requests keeps the number of sent requests significantly lower compared to the basic algorithm.

### 4.4.5 Evaluation

Our evaluation investigates the proposed geographic clustering algorithm as well as the two services for it. In Figure 4.19 we show the used Manhattan grid scenario. While we performed clustering and service provision for five intersections ($j_0$–$j_4$), we only collected results for the middle intersection $j_0$. We did this to avoid border effects introduced by being the last cluster in any direction. Furthermore, the road

traffic was not limited to the depicted area, but cars were driving in a much larger area. This was also done to minimize effects triggered by the scenario edges.

To better understand the clustering process and the performance of the services we used three different traffic densities. For the lowest density, it was not always possible to create a cluster resulting in a loss of data. The mean number of cars at $j_0$ for the low density was 2 and rose to a maximum of 8 cars. For medium and large densities, the cluster was usually large enough to sustain a cluster at a specific location and in turn losing less data. On average, the medium density lead to 4 cars with a maximum of 17 cars, while the high traffic density resulted in an average cluster size of 9 with a peak of 80 cars.

We detail the simulation parameters in their respective section. Nevertheless, some are valid for all configurations and can be found in Table 4.3. To acquire statistically relevant results, we performed a large number of repetitions. As these numbers differ from configuration to configuration, the exact numbers can also be found in the specific section. Note that, error bars in the plots indicate ± the standard deviation.

#### 4.4.5.1 Cluster Structure

To investigate the geographic clustering algorithm, we first focus on the cluster structure. We do this by looking at how long a car takes the role of CH or CM. The results of this can be found in Figure 4.20 and the specific simulation parameters in Table 4.4. As can be seen the evaluation was done for all three traffic densities.

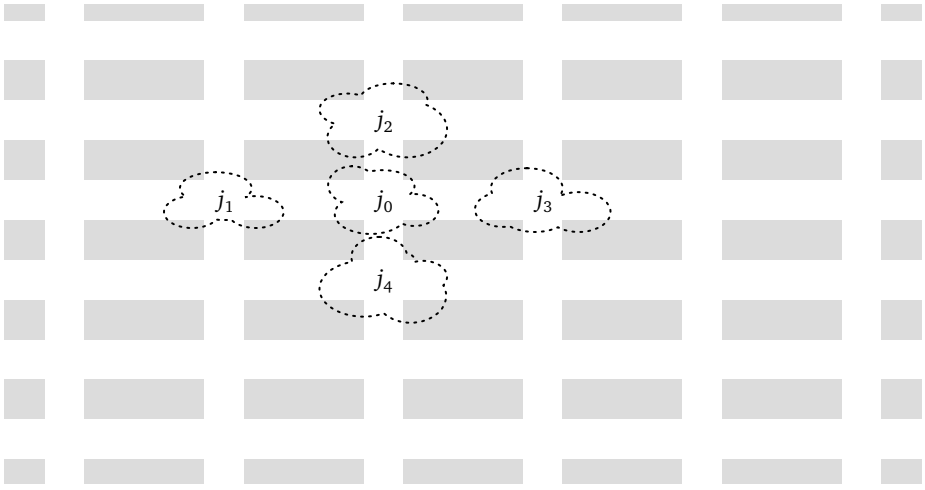


**Figure 4.19** – An overview of the used Manhattan grid scenario. All metrics were collected for the cluster on junction $j_0$. The other intersections $j_{1-4}$ participated in the clustering process, but were not considered for the evaluation; based on [57]

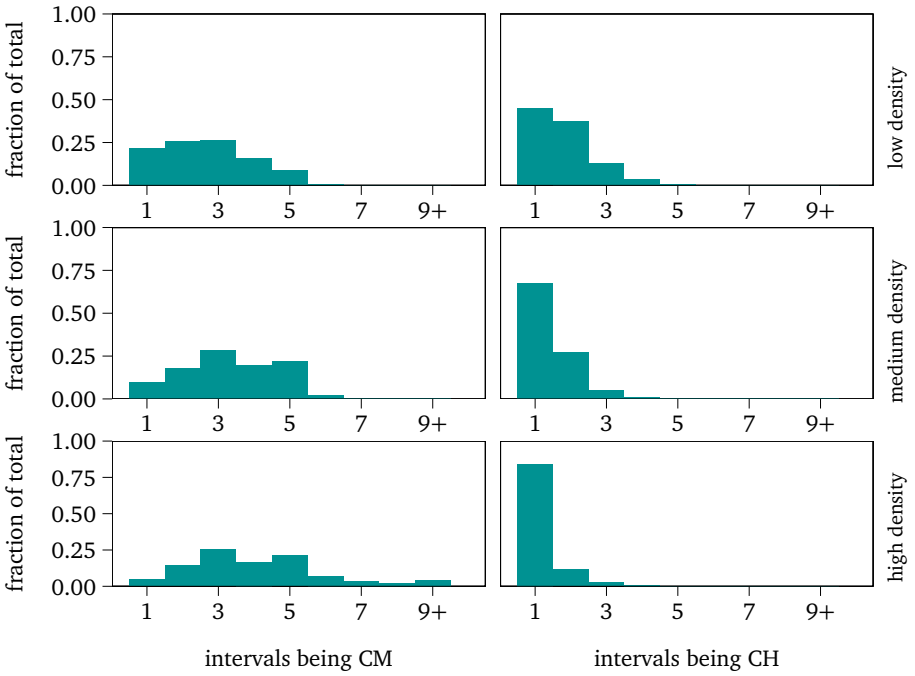


**Figure 4.20** – Histograms showing how long a car takes the role of CM or a CH. Parameters are the three traffic densities; based on [57]

**Table 4.3** – General geographic clustering simulation parameters.

| 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) |

**Table 4.4** – Cluster structure simulation parameters.

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

Generally, cars are longer a CM compared to being a CH. This is expected as only a single car can be CH, but multiple cars are CMs in parallel. The same reasoning applies when looking how long a car is CH in the three densities. As there are more potential candidates in a scenario with a high density, the CH changes more often resulting in less consecutive intervals. Higher traffic leads to more and longer queues at intersections. Therefore, cars are longer part of a cluster, i.e., they are CM for more intervals. We observe this effect in the first column where cars are longer part of a cluster when the density increases. While for low traffic density cars are mostly part of a cluster for 1–3 intervals, for a medium density the range increases to 3–5. For the highest density, cars can be part of a cluster for an even longer time.

#### 4.4.5.2 Collect Service

The main issue regarding the *collect* service is lost data in the last hop, i.e., when sending from CH to the AP [56]. As the aggregated data of a whole cluster is sent, losing it would result in a much higher loss compared to losing data between a CM and its CH. By looking at the results from our latest scenario we can confirm this even for multiple traffic densities. The specific simulation configurations for the *collect* service can be found in Table 4.5.

For evaluating the *collect* service, we focused on the *success rate*, i.e., the number of received compared to the number of generated data. It is defined by $r/g$, where $r$ is the data received by the AP and $g = b \times |\text{CM}|$ the generated data by all CMs. Beside the traffic density, we further investigated the results using two parameters:

- **Aggregation Factor**: As outlined the CH potentially processes and aggregates the data it receives from CMs. Afterwards, the data is sent to the AP. If we have an aggregation factor of 0.4, the data from the KB was reduced to 40 % before being sent.

- **Retransmissions**: The basic algorithm sends all messages, even the data, via broadcast. Therefore, no retransmissions are available, hurting especially the already mentioned last transmission. To avoid losing data, we enable unicast transmissions for data messages. The used algorithm follows the IEEE 802.11p

**Table 4.5** – Collect service simulation parameters.

| Parameter | Value |
|---|---|
| CM to CH data | 10 kB 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 |

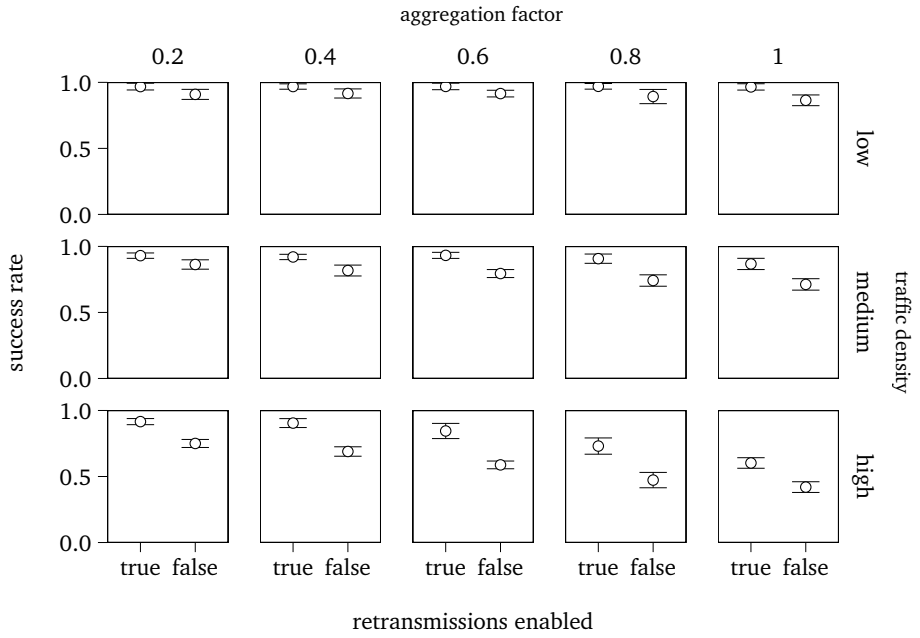


**Figure 4.21** – Success of the *collect* service. Parameters are enabled and disabled retransmissions, multiple aggregation factors, and the three traffic densities; based on [57]

mechanism and tries to retransmit every packet up to 7 times. Note that, this unicast mechanism has not been adapted ideally for vehicular networks [164] but still works as a proof of concept.

Figure 4.21 depicts the mean *success rate* for the outlined configurations. If there is not much traffic, the *collect* service works very well. By enabling retransmissions, close to 100 % of the generated data is received by the AP no matter the aggregation factor. Even without them, the success is above 90 %. A slightly worse result can be seen when looking at the medium traffic density, i.e., the middle row. Again, retransmissions improve the performance to values above 90 %. Otherwise, the success rate ranges from roughly 75 % to 90 % depending on the aggregation. When looking at the last row, which shows the high traffic density results, we can see an even larger performance decrease. While retransmissions still improve the performance, without aggregation the success is only slightly above 50 %, i.e., nearly half of the data gets lost. But, if it is possible to aggregate the data, the success rate greatly increases to approximately 90 %.

Now we wanted to investigate why the performance degrades if no aggregation is applied. For this we looked at the number of transmission errors on the receiver side for all three traffic densities and an aggregation factor of 1. Such an error happens if the wireless frame cannot be decoded, e.g., if there is still a transmission ongoing or

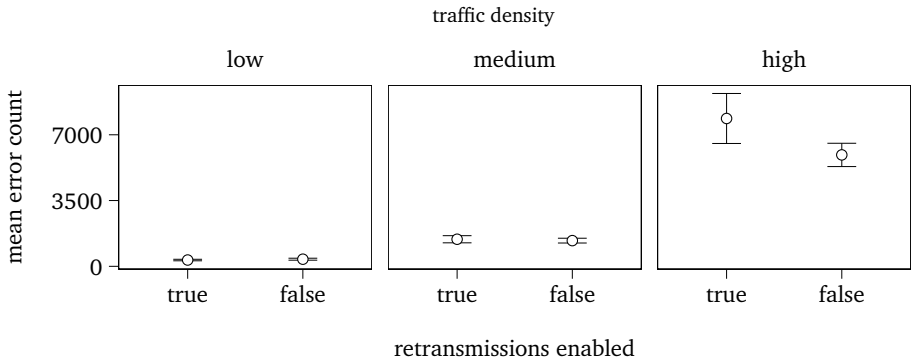


**Figure 4.22** – Number of received erroneous frames when using the *collect* service for an aggregation factor of 1. Parameters are enabled and disabled retransmissions and the three traffic densities; based on [57]

a bit error occurred during the transmission. The results can be seen in Figure 4.22. For low and medium traffic density, there are barely any errors. More interestingly there is no difference between having retransmissions enabled or not. But, when looking at the high traffic density, a much larger number of errors occurs, which results in a worse performance. As the packets sent from a CH to the AP are larger compared to the ones from CMs to CH, they are more likely to fail. Nevertheless, due to retransmissions, more data is successfully received (cf. Figure 4.21).

#### 4.4.5.3 Preserve Service

For our evaluation of the *preserve* service, we investigated the *fraction of known data* and the *channel load*. First, the *fraction of known data* shows how well the service itself performs. Based on the overall known data and the actually known data, this metric describes how much data has been lost. To acquire the data, the AP keeps a record of how much data should be known (by at least a single car) in the cluster. For a data fragment to qualify for overall known data, at least one CM has to notify the AP that is stored the fragment and the TTL has not expired. As the AP is aware which data is currently available in the cluster, it can produce the *fraction of known*

**Table 4.6** – Preserve service simulation parameters.

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

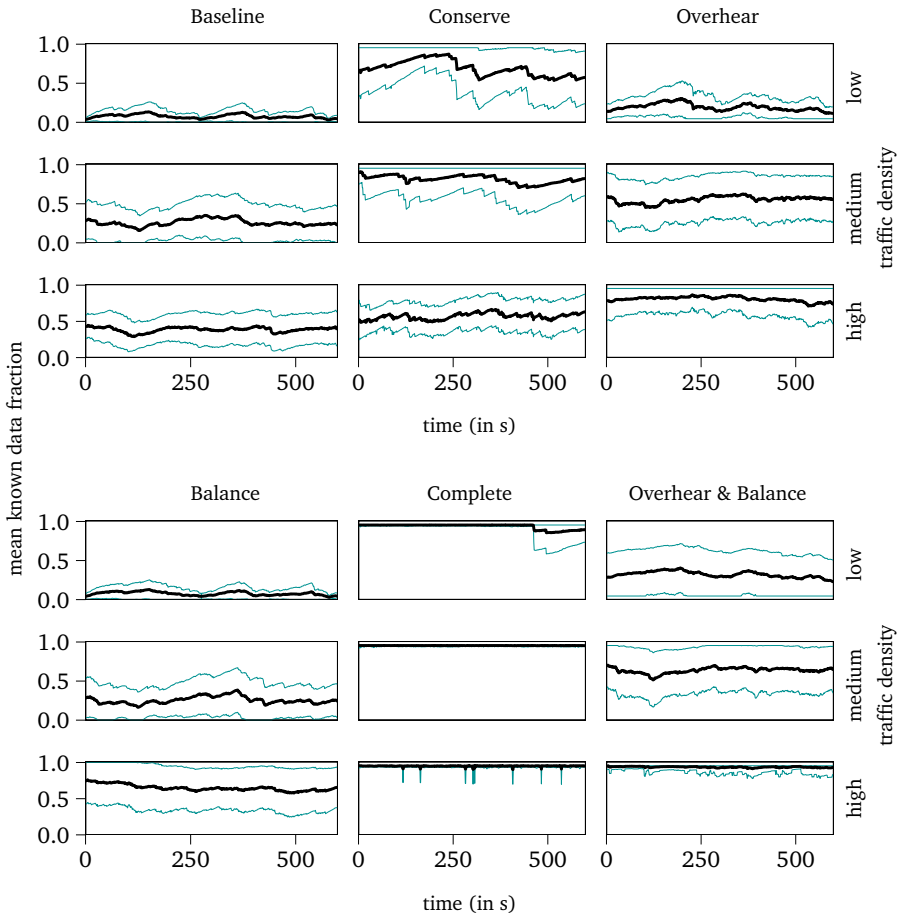


**Figure 4.23** – The success rate of keeping data available in the cloud over time when using the *preserve* service. Parameters are the different improvements and the three traffic densities. Note that the black line shows the mean and the turquoise area the standard deviation; based on [57]

*data*. This metric shows how well data is preserved inside the cluster. Second, the *channel load* shows the effects of the application on the wireless channel. It is based on the MAC busy fraction as observed by the AP. Furthermore, it often explains worse application performance and points towards solutions.

To explore the performance of the *preserve* application we used a TTL of 300 s. By keeping data valid for 5 min, cars store their data fragments in multiple micro clouds they pass by. Therefore, data is kept inside the cloud even after the car left. Furthermore, for each density we had 10 configurations. While the overall density stayed the same, the different configurations allowed a reduction of simulation artifacts triggered by using only a single scenario. Further specific simulation configurations can be found in Table 4.6.

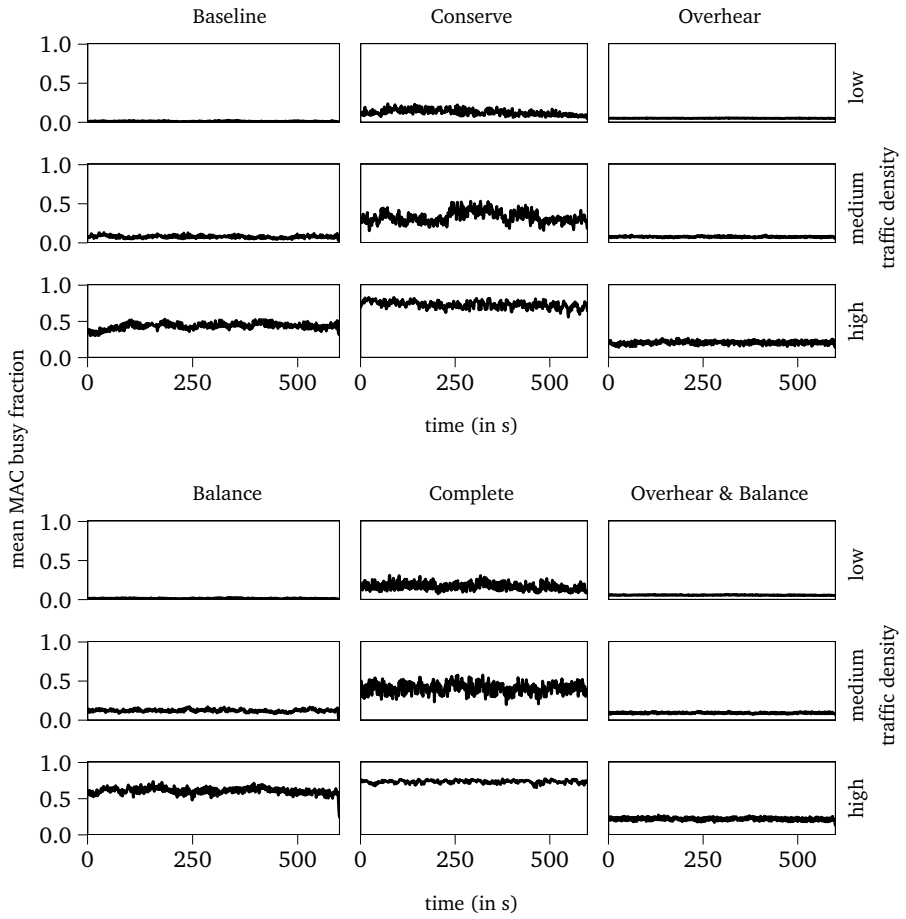


**Figure 4.24** – The mean channel load over time when using the *preserve* service. Parameters are the different improvements and the three traffic densities; based on [57]

The results outlining the *fraction of known data* are shown in Figure 4.23. They are accompanied by the *channel load* results in Figure 4.24. In the first column of the first row we can see the baseline without any of the discussed improvements. The low-density scenarios are barely able to preserve any data. This is due to regularly having no cars available for clustering or a cluster with only a single car, also leading to a total loss of all data. Medium and high density are slightly better, but still the fraction of known data is below 50 %. Here, losing all data still happens, but is less likely. Note that, even a small cluster with 1 or 2 CMs or a complete exchange of CMs is prone to loss of data fragments.

Enabling the *Conserve* improvement already leads to significantly better results, especially for low and medium density scenarios. Data from other clusters is not deleted anymore, but actively shared. Therefore, even if the cluster is empty for some time, the chance of reacquiring data is higher. The only small performance

improvements of the high-density plot can be explained by looking at the channel load. It is significantly higher compared to the baseline and to the other densities. Therefore, the service is not able to exchange data.

The next enabled improvement is the *Overhear* enhancement. If there are only a few cars, there is nothing to overhear. This is especially visible when looking at the low-density scenarios results, but also true for the medium one. But, for the high-density scenario, there is a significant performance improvement visible. Due to the large number of cars in these scenarios, there are many potential recipients for data fragments, which leads to a high fraction of known data. Moreover, as overhearing leads to less data being requested, the load on the channel with this improvement is considerably better, i.e., lower.

Slightly worse results can be observed with only the *Balance* improvement enabled (shown in the first column of the second row). Still, it is more effective compared to the baseline, particularly when looking at high density scenarios. For other densities, this improvement does not yield any significant improvements as the number of missing data fragments is rather low and balancing does not have much influence.

The second column of the second row shows all three *preserve* service improvements enabled. Results indicate a good performance with the average known data fraction being close to 100 %, independent of the density. Permanent loss of all data is a very rare event as the combination enables fast recovery in such cases. Unfortunately for high density scenarios, the good performance induces a high channel load. This leaves barely any space on the wireless channel for other applications. The reason for this is the *Conserve* improvement, which, while barely improving the high-density performance, incurs the high load. This brings us to the bottom last column where we enabled the *Overhear* and the *Balance* improvements together but disabled the *Conserve* improvement. Compared to having all improvements enabled, the fraction of known data for the low and medium density scenarios is reduced. But, the performance for the high-density scenarios is similar while reducing the channel load significantly (below 30 %).

We can conclude that all three improvements should be enabled to increase the *preserve* service performance. Nevertheless, the AP should closely monitor the channel load and if it gets too high, disable the *conserve* improvement. This would yield a similar performance, but with a significantly reduced channel load, which in turn enables more applications.

### 4.4.6 Lessons Learned

In this section we presented an approach to form micro clouds at specific geographic locations. Usually located at intersections, we relied on a coordinator node to calculate the cluster configuration using moving cars. Such a coordinator is envisioned to

be existing infrastructure such as Wi-Fi APs, IEEE 802.11p RSUs or LTE eNBs. Beside forming the micro clouds themselves, we also proposed two services to make them more useful. First, the *collect* service aggregates data from all CMs and uploads this data via the coordinator to the macro cloud. Second, the *preserve* service keeps local data available in the cloud in order to offload the macro cloud.

While evaluating the *collect* service, we discovered that the last link, i.e., from CH to AP, is the most important one. If data gets lost there, information from a whole cluster is lost compared to only data from a single CM if the transmission between CM and CH fails. To avoid such issues, we propose to use retransmissions, which already improves the performance. Still, there is potential to make this even better, e.g., by relying on a second technology for transmissions to the AP, or making the link even more resilient by introducing additional retransmissions.

For the *preserve* service we propose multiple improvements to keep the data for as long as possible. We discovered that, depending on the traffic density, the effects strongly vary. If all are enabled, the data is kept consistently for all densities, but incurs a large load on the channel for the highest density. Therefore, we disabled an improvement and discovered that this still works well for high densities while significantly reducing the channel load. This indicates further research regarding an adaptive approach could improve the *preserve* service even more. Such an approach would disable and enable improvements based on the channel load and therefore would work for varying traffic densities without incurring a high channel load.

# Chapter 5

# Conclusion

In this Ph.D. thesis, we investigated the idea of using connected cars as key building blocks for future smart cities. Such cities are usually defined by certain characteristics, which can be divided into building blocks and goals. While the former include Information and Communication Technology (ICT), mobility, the natural environment, and the cities inhabitants, goals are for example a more efficient government, prosperous business, and happier citizens. As connected cars are essentially a combination of two of these building blocks (ICT and mobility), we consider them to be a natural fit for service provision in smart cities. Therefore, in the scope of this thesis, we proposed multiple architectures with connected cars as the center piece.

An initial investigation of smart city definitions revealed that connected cars indeed match characteristics commonly associated with smart cities. Furthermore, we studied technologies and use cases for connected cars and outlined a primary service category for this thesis, i.e., *Virtual Infrastructure*. In this context, connected cars form a vehicular network to replace or complement networking infrastructure.

Our first contribution was the *Car4ICT* architecture where we relied on connected cars to support users in discovering and utilizing arbitrary services, e.g., multimedia file sharing, storage offloading, and automated weather forecasts. We outlined involved entities, access procedures, and how to identify and search for offered services. When evaluating the architecture, we first focused on urban environments reflecting a smart city. We found that our proposed service discovery works well, even with a relatively low penetration rate of participating cars. Only in scenarios with a very low number of connected cars, a user would experience worse quality induced by long delays. Afterwards, we extended the architecture to freeways to make services farther away available to users. For this, we adapted the architecture to support Contention Based Forwarding (CBF) and Store-Carry-Forward and evaluated the changes using a realistic freeway scenario. Hereby, we discovered that the CBF parameters proposed by the European Telecommunications Standards Institute

can be adapted to reduce the transfer delay over distances up to 100 km. Like in urban environments, the penetration rate was also a challenge, which induced a significantly longer delay.

Our second contribution was to group connected cars to form Vehicular Clouds (VCs). This provides multiple opportunities, e.g., resource offloading, service provision, and replacing or complementing network infrastructure. Generally, infrastructure can be a solution to the encountered issues with low penetration rates. As building additional infrastructure can be expensive, parked cars and their unused resources are a cheaper option. Therefore, our goal was to rely on parked cars to form VCs providing additional virtual infrastructure. Beside forming clouds, we selected a subset of the participating cars, so called gateways, to be open for connections from outside users. The goal was to reduce the load on the wireless channel while barely impacting data transfers between users and the cloud. Furthermore, by connecting to multiple of these gateways after another, users can transfer larger amounts of data. To achieve this, we presented a handover mechanism. Our evaluation of these concepts showed that gateway selection indeed reduces the channel load while lowering the amount of transferred data only slightly. Additionally, we found out that the penetration rate of connected cars trying to connect to the VC had a different effect compared to the previously presented Car4ICT architecture. In the context of these VCs, a low rate led to an improved performance while a high rate did the opposite. The reason for this was a high channel load leading to higher delays and subsequently lost data as users moved on. As a future research direction, an adaptive approach might be able to mitigate such issues related to penetration rate and channel load.

As our third contribution, we formed VCs at specific geographic locations using moving cars. Beside the clustering algorithm, we proposed two core data management services showing the potential of these clouds. The first of these services collects aggregated data from participating cars and provides it via an uplink to a data center where it subsequently can be used, e.g., to improve the flow of traffic. The second data management service preserves locally relevant data for a longer time and by doing this offloads resources from the data center or the internet. We evaluated both services with varying vehicle densities in an urban scenario. Again, with a high traffic density, i.e., many equipped vehicles, the performance degrades due to load on the wireless channel. The other way around, for low traffic densities, the cloud tends to lose offloaded data when running the preserving service. To solve both these issues we proposed multiple improvements and evaluated them. As a result, we were able to significantly improve the performance for all traffic densities. Hereby, we again argue that an adaptive algorithm — activating and deactivating these improvements based on traffic density — is an interesting future direction.

Two challenges we faced when devising our solutions are also reflected in the literature: low penetration rates and high channel load. If the load is too high, the proposed architectures will suffer from a degrading performance and no other applications and services can be run in parallel. While we were able to provide solutions for some of these challenges, there certainly exist further approaches. One example would be to make the architectures more adaptive and in turn reduce access to them. This would reduce the number of parallel users, but could provide better quality to those connected. To avoid issues related to low penetration rates, relying on infrastructure might be the way to go. This could either be in the form of the proposed parked cars or by utilizing cellular networks. Generally, the architectures proposed in this thesis are technology independent and it is easy to adapt them for LTE, 5G, and future Wireless LAN (WLAN) standards.

Right now, connected cars are already available from multiple manufacturers. Still, they can be considered to be in the early introductory phase and are not yet ready to provide services as presented in this thesis. Nevertheless, we were able to show that they indeed can be considered a candidate for being a key smart city enabler. The architectures proposed throughout this thesis show the potential of connected cars and can be seen as a foundation for future research. Beside the highlighted challenges, next steps towards making connected cars the envisioned smart city building block include small-scale experiments and studying them in the context of future networking standards like Cellular V2X.

# List of Abbreviations

| | |
|---|---|
| **AP** | Access Point |
| **ARQ** | Automatic Repeat Request |
| **C-V2X** | Cellular V2X |
| **CAM** | Cooperative Awareness Message |
| **CBF** | Contention Based Forwarding |
| **CH** | Cluster Head |
| **CM** | Cluster Member |
| **D2D** | Device-to-Device |
| **DHT** | Distributed Hash Table |
| **DSRC** | Dedicated Short-Range Communication |
| **eNB** | eNodeB |
| **ETSI** | European Telecommunications Standards Institute |
| **FCD** | Floating Car Data |
| **GNSS** | Global Navigation Satellite System |
| **ICN** | Information-Centric Networking |
| **ICT** | Information and Communication Technology |
| **IoT** | internet of Things |
| **IVC** | Inter-Vehicle Communication |
| **KB** | Knowledge Base |
| **M2M** | Machine-to-Machine |
| **MAC** | Medium Access Control |
| **MANET** | Mobile Ad Hoc Network |
| **MEC** | Mobile Edge Computing |
| **MMC** | Micro-Macro Cloud |
| **NDN** | Named Data Networking |
| **NIC** | Network Interface Controller |
| **O/D** | Origin-Destination |
| **PHY** | Physical layer |
| **PKI** | Public Key Infrastructure |
| **POI** | Point Of Interest |

| | |
|---|---|
| **RSU** | Roadside Unit |
| **SCF** | Store-Carry-Forward |
| **SDP** | Service Discovery Protocol |
| **SINR** | Signal to Interference and Noise Ratio |
| **TIS** | Traffic Information System |
| **TTL** | Time to Live |
| **V2I** | Vehicle-to-Infrastructure |
| **V2V** | Vehicle-to-Vehicle |
| **VANET** | Vehicular Ad Hoc Network |
| **VC** | Vehicular Cloud |
| **VCP** | Virtual Cord Protocol |
| **VLC** | Visible-Light Communication |
| **VTL** | Virtual Traffic Light |
| **WAVE** | Wireless Access in Vehicular Environments |
| **WLAN** | Wireless LAN |

# List of Figures

# List of Tables

# Bibliography


[1] United Nations, "World Urbanization Prospects: The 2018 Revision [key facts]," United Nations, Tech. Rep., 2018.

[2] C. Yin, Z. Xiong, H. Chen, J. Wang, D. Cooper, and B. David, "A literature survey on smart cities," *Science China Information Sciences*, vol. 58, no. 10, Oct. 2015.

[3] W. Rong, Z. Xiong, D. Cooper, C. Li, and H. Sheng, "Smart City Architecture: A Technology Guide for Implementation and Design Challenges," *IEEE China Communications*, vol. 11, no. 3, pp. 56–69, Mar. 2014.

[4] V. Albino, U. Berardi, and R. M. Dangelico, "Smart Cities: Definitions, Dimensions, Performance, and Initiatives," *Routledge Journal of Urban Technology*, vol. 22, no. 1, pp. 3–21, Feb. 2015.

[5] R. G. Hollands, "Will the real smart city please stand up?" *Routledge City*, vol. 12, no. 3, 303–320, Nov. 2008.

[6] A. Caragliu, C. Del Bo, and P. Nijkamp, "Smart Cities in Europe," *Routledge Journal of Urban Technology*, vol. 18, no. 2, pp. 65–82, Aug. 2011.

[7] R. Giffinger and G. Haindlmaier, "Smart Cities Ranking: An Effective Instrument for the Positioning of the Cities?" *ACE: Architecture, City, and Environment*, vol. 4, no. 12, 7–26, Feb. 2010.

[8] C. Harrison, B. Eckman, R. Hamilton, P. Hartswick, J. Kalagnanam, J. Paraszczak, and P. Williams, "Foundations for Smarter Cities," *IBM Journal of Research and Development*, vol. 54, no. 4, pp. 1–16, Jul. 2010.

[9] T. Nam and T. A. Pardo, "Conceptualizing Smart City with Dimensions of Technology, People, and Institutions," in *12th Annual International Digital Government Research Conference: Digital Government Innovation in Challenging Times*, College Park, MD: ACM, Jun. 2011, 282–291.

[10] T. Nam and T. A. Pardo, "Smart City as Urban Innovation: Focusing on Management, Policy, and Context," in *5th International Conference on Theory and Practice of Electronic Governance*, Tallinn, Estonia: ACM, Sep. 2011, 185–194.

[11] C. Benevolo, R. P. Dameri, and B. D'Auria, "Smart Mobility in Smart City," in *Empowering Organizations*, ser. Lecture Notes in Information Systems and Organisation, T. Torre, A. M. Braccini, and R. Spinelli, Eds., vol. 11, Genoa, Italy: Springer International Publishing, Nov. 2016, 13–28.

[12] M. Seredynski and P. Bouvry, "A Survey of Vehicular-based Cooperative Traffic Information Systems," in *14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, Washington, D.C.: IEEE, Oct. 2011, pp. 163–168.

[13] M. Ferreira, R. Fernandes, H. Conceição, W. Viriyasitavat, and O. K. Tonguz, "Self-organized Traffic Control," in *7th ACM International Workshop on Vehicular Internetworking (VANET 2010)*, Chicago, IL: ACM, Sep. 2010, pp. 85–90.

[14] M. Ferreira and P. d'Orey, "On the Impact of Virtual Traffic Lights on Carbon Emissions Mitigation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 1, pp. 284–295, 2012.

[15] Toyota. (Nov. 2014). Toyota to Bring Vehicle-Infrastructure Cooperative Systems to New Models in 2015. Press Release, [Online]. Available: `https://newsroom.toyota.co.jp/en/detail/4228471/`.

[16] General Motors. (Mar. 2017). V2V Safety Technology Now Standard on Cadillac CTS Sedans. Press Release, [Online]. Available: `https://media.cadillac.com/media/us/en/cadillac/news.detail.html/content/Pages/news/us/en/2017/mar/0309-v2v.html`.

[17] Volkswagen. (Feb. 2018). Volkswagen Group assumes pioneering role in rapid road safety improvement. Press Release, [Online]. Available: `https://www.volkswagenag.com/en/news/2018/02/volkswagen_group_rapid_road_safety.html`.

[18] M. Segata, B. Bloessl, S. Joerer, C. Sommer, M. Gerla, R. Lo Cigno, and F. Dressler, "Towards Communication Strategies for Platooning: Simulative and Experimental Evaluation," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 12, pp. 5411–5423, Dec. 2015.

[19] T. L. Willke, P. Tientrakool, and N. F. Maxemchuk, "A Survey of Inter-Vehicle Communication Protocols and Their Applications," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 2, pp. 3–20, Jun. 2009.

[20] K. Zheng, Q. Zheng, P. Chatzimisios, W. Xiang, and Y. Zhou, "Heterogeneous Vehicular Networking: A Survey on Architecture, Challenges, and Solutions," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2377–2396, Jun. 2015.

[21] G. Karagiannis, O. Altintas, E. Ekici, G. Heijenk, B. Jarupan, K. Lin, and T. Weil, "Vehicular Networking: A Survey and Tutorial on Requirements, Architectures, Challenges, Standards and Solutions," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 4, pp. 584–616, Nov. 2011.

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

[23] S. Joerer, M. Segata, B. Bloessl, R. Lo Cigno, C. Sommer, and F. Dressler, "A Vehicular Networking Perspective on Estimating Vehicle Collision Probability at Intersections," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 4, pp. 1802–1812, May 2014.

[24] X. Yang, J. Liu, F. Zhao, and N. Vaidya, "A Vehicle-to-Vehicle Communication Protocol for Cooperative Collision Warning," in *The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MOBIQUITOUS 2004)*, Boston, MA: IEEE, Aug. 2004, pp. 114–123.

[25] W. Viriyasitavat and O. Tonguz, "Priority Management of Emergency Vehicles at Intersections Using Self-Organized Traffic Control," in *Vehicular Technology Conference (VTC2012-Fall)*, Quebec City, Canada: IEEE, Sep. 2012.

[26] European Commission. (Apr. 2015). eCall in all new cars from April 2018, [Online]. Available: `https://ec.europa.eu/digital-single-market/news/ecall-all-new-cars-april-2018`.

[27] C. Sommer, O. K. Tonguz, and F. Dressler, "Traffic Information Systems: Efficient Message Dissemination via Adaptive Beaconing," *IEEE Communications Magazine*, vol. 49, no. 5, pp. 173–179, May 2011.

[28] L. Wischhof, A. Ebner, and H. Rohling, "Information Dissemination in Self-Organizing Intervehicle Networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 1, pp. 90–101, Mar. 2005.

[29] IEEE, "IEEE Standard for Wireless Access in Vehicular Environments (WAVE) - Over-the-Air Electronic Payment Data Exchange Protocol for Intelligent Transportation Systems (ITS)," IEEE, Std 1609.11-2010, Jan. 2011.

[30] F. Malandrino, C. Casetti, C.-F. Chiasserini, C. Sommer, and F. Dressler, "The Role of Parked Cars in Content Downloading for Vehicular Networks," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 9, pp. 4606–4617, Nov. 2014.

[31] R. Baldessari, B. Bödekker, A. Brakemeier, M. Deegener, A. Festag, W. Franz, A. Hiller, C. Kellum, T. Kosch, A. Kovacs, et al., "CAR 2 CAR Communication Consortium Manifesto," CAR 2 CAR Communication Consortium, Tech. Rep. 1.1, Aug. 2007.

[32] M. Gerla, "Vehicular Cloud Computing," in *11th IFIP/IEEE Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net 2012)*, Ayia Napa, Cyprus: IEEE, Jun. 2012, pp. 152–155.

[33] S. Olariu, T. Hristov, and G. Yan, "The Next Paradigm Shift: From Vehicular Networks to Vehicular Clouds," in *Mobile Ad Hoc Networking*, Wiley, 2013, pp. 645–700.

[34] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile Edge Computing - A key technology towards 5G," ETSI, White Paper No. 11, Sep. 2015.

[35] P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, Mar. 2017.

[36] T. Higuchi, J. Joy, F. Dressler, M. Gerla, and O. Altintas, "On the Feasibility of Vehicular Micro Clouds," in *9th IEEE Vehicular Networking Conference (VNC 2017)*, Torino, Italy: IEEE, Nov. 2017, pp. 179–182.

[37] P. K. Agyapong, M. Iwamura, D. Staehle, W. Kiess, and A. Benjebbour, "Design Considerations for a 5G Network Architecture," *IEEE Communications Magazine*, vol. 52, no. 11, pp. 65–75, Nov. 2014.

[38] B. Bangerter, S. Talwar, R. Arefi, and K. Stewart, "Networks and Devices for the 5G Era," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 90–96, Feb. 2014.

[39] A. Osseiran, F. Boccardi, V. Braun, K. Kusume, P. Marsch, M. Maternia, O. Queseth, M. Schellmann, H. Schotten, H. Taoka, et al., "Scenarios for 5G Mobile and Wireless Communications: the Vision of the METIS Project," *IEEE Communications Magazine*, vol. 52, no. 5, pp. 26–35, May 2014.

[40] European Commision, "Broadband Coverage in Europe 2017," European Commision, Study, Jul. 2017.

[41] "Final Report of Subproject 5," simTD Project, Deliverable D5.5 Part A, Jun. 2013.

[42] M. M. Afsar and M.-H. Tayarani-N, "Clustering in sensor networks: A literature survey," *Elsevier Journal of Network and Computer Applications*, vol. 46, pp. 198–226, Nov. 2014.

[43] C. Sommer, D. Eckhoff, R. German, and F. Dressler, "A Computationally Inexpensive Empirical Model of IEEE 802.11p Radio Shadowing in Urban Environments," in *8th IEEE/IFIP Conference on Wireless On demand Network Systems and Services (WONS 2011)*, Bardonecchia, Italy: IEEE, Jan. 2011, pp. 84–90.

[44] C. Cooper, D. Franklin, M. Ros, F. Safaei, and M. Abolhasan, "A Comparative Survey of VANET Clustering Techniques," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 657–681, Feb. 2017.

[45] T. Higuchi, F. Dressler, and O. Altintas, "How to Keep a Vehicular Micro Cloud Intact," in *87th IEEE Vehicular Technology Conference (VTC2018-Spring)*, Porto, Portugal: IEEE, Jun. 2018.

[46] A. Papathanassiou and A. Khoryaev, "Cellular V2X as the Essential Enabler of Superior Global Connected Transportation Services," *IEEE 5G Tech Focus*, vol. 1, no. 2, Jun. 2017.

[47] IEEE, "Wireless Access in Vehicular Environments," IEEE, Std 802.11p-2010, Jul. 2010.

[48] C. B. Liu, B. Sadeghi, and E. W. Knightly, "Enabling Vehicular Visible Light Communication (V2LC) Networks," in *8th ACM International Workshop on Vehicular Internetworking (VANET 2011)*, Las Vegas, NV: ACM, Sep. 2011, 41–50.

[49] F. Hagenauer, F. Dressler, and C. Sommer, "A Simulator for Heterogeneous Vehicular Networks," in *6th IEEE Vehicular Networking Conference (VNC 2014), Poster Session*, Paderborn, Germany: IEEE, Dec. 2014, pp. 185–186.

[50] O. Altintas, F. Dressler, F. Hagenauer, M. Matsumoto, M. Sepulcre, and C. Sommer, "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)*, Hong Kong, China: IEEE, Apr. 2015, pp. 654–659.

[51] F. Hagenauer, C. Sommer, R. Onishi, M. Wilhelm, F. Dressler, and O. Altintas, "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)*, San Francisco, CA: IEEE, Apr. 2016, pp. 788–793.

[52] F. Hagenauer, F. Dressler, O. Altintas, and C. Sommer, "Cars as a Main ICT Resource of Smart Cities," in *Smart Cities and Homes - Key Enabling Technologies*, M. S. Obaidat and P. Nicopolitidis, Eds., Elsevier, May 2016, pp. 131–147.

[53] F. Hagenauer, C. Sommer, T. Higuchi, O. Altintas, and F. Dressler, "Using Clusters of Parked Cars as Virtual Vehicular Network Infrastructure," in *8th IEEE Vehicular Networking Conference (VNC 2016), Poster Session*, Columbus, OH: IEEE, Dec. 2016, pp. 126–127.

[54] F. Hagenauer, C. Sommer, T. Higuchi, O. Altintas, and F. Dressler, "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)*, Snowbird, UT: ACM, Oct. 2017, pp. 57–64.

[55] F. Hagenauer, C. Sommer, T. Higuchi, O. Altintas, and F. Dressler, "Vehicular Micro Cloud in Action: On Gateway Selection and Gateway Handovers," *Elsevier Ad Hoc Networks*, vol. 78, pp. 73–83, Sep. 2018.

[56] F. Hagenauer, C. Sommer, T. Higuchi, O. Altintas, and F. 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)*, Snowbird, UT: ACM, Oct. 2017, pp. 31–35.

[57] F. Hagenauer, T. Higuchi, O. Altintas, and F. Dressler, "Efficient Data Handling in Vehicular Micro Clouds," *Elsevier Ad Hoc Networks*, vol. 91, p. 101 871, Aug. 2019.

[58] F. Hagenauer, P. Baldemaier, F. Dressler, and C. Sommer, "Advanced Leader Election for Virtual Traffic Lights," *ZTE Communications, Special Issue on VANET*, vol. 12, no. 1, pp. 11–16, Mar. 2014.

[59] F. Hagenauer, C. Sommer, S. Merschjohann, T. Higuchi, F. Dressler, and O. Altintas, "Cars as the Base for Service Discovery and Provision in Highly Dynamic Networks," in *35th IEEE Conference on Computer Communications (INFOCOM 2016), Demo Session*, San Francisco, CA: IEEE, Apr. 2016, pp. 358–359.

[60] A. Memedi, F. Hagenauer, F. Dressler, and C. Sommer, "Cluster-based Transmit Power Control in Heterogeneous Vehicular Networks," in *7th IEEE Vehicular Networking Conference (VNC 2015)*, Kyoto, Japan: IEEE, Dec. 2015, pp. 60–63.

[61] M. Mutschlechner, F. Klingler, F. Erlacher, F. Hagenauer, M. Kiessling, and F. Dressler, "Reliable Communication using Erasure Codes for Monitoring Bats in the Wild," in *33rd IEEE Conference on Computer Communications*

*(INFOCOM 2014), Student Activities*, Toronto, Canada: IEEE, Apr. 2014, pp. 189–190.

[62] C. Sommer, F. Hagenauer, and F. Dressler, "A Networking Perspective on Self-Organizing Intersection Management," in *IEEE World Forum on Internet of Things (WF-IoT 2014)*, Seoul: IEEE, Mar. 2014, pp. 230–234.

[63] F. Dressler, G. S. Pannu, F. Hagenauer, M. Gerla, T. Higuchi, and O. Altintas, "Virtual Edge Computing Using Vehicular Micro Clouds," in *IEEE International Conference on Computing, Networking and Communications (ICNC 2019)*, Honolulu, HI: IEEE, Feb. 2019.

[64] C. Sommer, D. Eckhoff, A. Brummer, D. S. Buse, F. Hagenauer, S. Joerer, and M. Segata, "Veins – the open source vehicular network simulation framework," in *Recent Advances in Network Simulation*, A. Virdis and M. Kirsche, Eds., to appear, Springer, 2019, pp. 1–1.

[65] M. Angelidou, "Smart cities: A conjuncture of four forces," *Elsevier Cities*, vol. 47, no. Current Research of Cities (CRoC), 95–106, Sep. 2015.

[66] IEEE, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," IEEE, Std 802.11-2012, 2012.

[67] IEEE, "IEEE Guide for Wireless Access in Vehicular Environments (WAVE) - Architecture," IEEE, Std 1609.0-2013, Mar. 2014.

[68] ETSI, "Intelligent Transport Systems (ITS); Decentralized Congestion Control Mechanisms for Intelligent Transport Systems operating in the 5 GHz range; Access layer part," ETSI, TS 102 687 V1.1.1, Jul. 2011.

[69] ARIB, "700 MHz Band Intelligent Transport Systems," English, ARIB, STD T109-v1.2, Dec. 2013.

[70] X. Lin, J. G. Andrews, A. Ghosh, and R. Ratasuk, "An Overview of 3GPP Device-to-Device Proximity Services," *IEEE Communications Magazine*, vol. 52, no. 4, pp. 40–48, Apr. 2014.

[71] ETSI, "Universal Mobile Telecommunications System (UMTS); LTE; Proximity-based services (ProSe); Stage 2," ETSI, TS 123 303 V13.2.0, Mar. 2016.

[72] R. Molina-Masegosa and J. Gozalvez, "LTE-V for Sidelink 5G V2X Vehicular Communications: A New 5G Technology for Short-Range Vehicle-to-Everything Communications," *IEEE Vehicular Technology Magazine*, vol. 12, no. 4, pp. 30–39, Dec. 2017.

[73] W. Bronzi, R. Frank, G. Castignani, and T. Engel, "Bluetooth Low Energy for Inter-Vehicular Communications," in *6th IEEE Vehicular Networking Conference (VNC 2014)*, Paderborn, Germany: IEEE, Dec. 2014, pp. 215–221.

[74] J. Hasch, E. Topak, R. Schnabel, T. Zwick, R. Weigel, and C. Waldschmidt, "Millimeter-Wave Technology for Automotive Radar Sensors in the 77 GHz Frequency Band," *IEEE Transactions on Microwave Theory and Techniques*, vol. 60, no. 3, pp. 845–860, Mar. 2012.

[75] C. Sommer and F. Dressler, Vehicular Networking. Cambridge University Press, Nov. 2014.

[76] A. Vinel, "3GPP LTE Versus IEEE 802.11p/WAVE: Which Technology is Able to Support Cooperative Vehicular Safety Applications?" *IEEE Wireless Communications Letters*, vol. 1, no. 2, pp. 125–128, Apr. 2012.

[77] D. Cavalcanti, D. Agrawal, C. Cordeiro, B. Xie, and A. Kumar, "Issues in Integrating Celluar Networks, WLANs, and MANETs: A Futuristic Heterogeneous Wireless Network," *IEEE Wireless Communications*, vol. 12, no. 3, pp. 30–41, 2005.

[78] B. B. Chen and M. C. Chan, "MobTorrent: A Framework for Mobile Internet Access from Vehicles," in *28th IEEE Conference on Computer Communications (INFOCOM 2009)*, Rio de Janeiro, Brazil: IEEE, Apr. 2009.

[79] G. Rémy, S.-M. Senouci, F. Jan, and Y. Gourhant, "LTE4V2X: LTE for a Centralized VANET Organization," in *IEEE Global Telecommunications Conference (GLOBECOM 2011)*, Houston, TX: IEEE, Dec. 2011.

[80] L.-C. Tung, J. Mena, M. Gerla, and C. Sommer, "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)*, Ajaccio, Corsica, France: IEEE, Jun. 2013.

[81] IEEE, "IEEE Standard for Wireless Access in Vehicular Environments (WAVE) - Multi-channel Operation," IEEE, Std 1609.4-2010, Feb. 2011.

[82] S. Joerer, M. Segata, B. Bloessl, R. Lo Cigno, C. Sommer, and F. Dressler, "To Crash or Not to Crash: Estimating its Likelihood and Potentials of Beacon-based IVC Systems," in *4th IEEE Vehicular Networking Conference (VNC 2012)*, Seoul, Korea: IEEE, Nov. 2012, pp. 25–32.

[83] ETSI, "Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service," ETSI, TS 102 637-2 V1.1.1, Apr. 2010.

[84] A. Davila and M. Nombela, "Platooning - Safe and Eco-Friendly Mobility," in *SAE 2012 World Congress & Exibition*, Detroit, Michigan: SAE, Apr. 2012.

[85] O. K. Tonguz and M. Boban, "Multiplayer games over Vehicular Ad Hoc Networks: A new application," *Elsevier Ad Hoc Networks*, vol. 8, no. 5, pp. 531–543, Jul. 2010.

[86] K. Mershad, H. Artail, and M. Gerla, "ROAMER: Roadside Units as message routers in VANETs," *Elsevier Ad Hoc Networks*, vol. 10, no. 3, pp. 479–496, Sep. 2012.

[87] C. Lochert, B. Scheuermann, C. Wewetzer, A. Luebke, and M. Mauve, "Data Aggregation and Roadside Unit Placement for a VANET Traffic Information System," in *5th ACM International Workshop on Vehicular Inter-Networking (VANET 2008)*, San Francisco, CA: ACM, Sep. 2008, pp. 58–65.

[88] D. Naboulsi and M. Fiore, "Characterizing the Instantaneous Connectivity of Large-Scale Urban Vehicular Networks," *IEEE Transactions on Mobile Computing*, vol. 16, no. 5, pp. 1272–1286, May 2017.

[89] F. Dressler, P. Handle, and C. 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)*, Philadelphia, PA: ACM, Aug. 2014, pp. 11–18.

[90] C. Sommer, D. Eckhoff, and F. Dressler, "IVC in Cities: Signal Attenuation by Buildings and How Parked Cars Can Improve the Situation," *IEEE Transactions on Mobile Computing*, vol. 13, no. 8, pp. 1733–1745, Aug. 2014.

[91] F. Dressler, H. Hartenstein, O. Altintas, and O. K. Tonguz, "Inter-Vehicle Communication – Quo Vadis," *IEEE Communications Magazine*, vol. 52, no. 6, pp. 170–177, Jun. 2014.

[92] A. Boukerche and R. E. De Grande, "Vehicular Cloud Computing: Architectures, Applications, and Mobility," *Elsevier Computer Networks*, vol. 135, pp. 171–189, Apr. 2018.

[93] M. Whaiduzzaman, M. Sookhak, A. Gani, and R. Buyya, "A survey on vehicular cloud computing," *Elsevier Journal of Network and Computer Applications*, vol. 40, pp. 325–344, Apr. 2014.

[94] S. Arif, S. Olariu, J. Wang, G. Yan, W. Yang, and I. Khalil, "Datacenter at the Airport: Reasoning about Time-Dependent Parking Lot Occupancy," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 11, pp. 2067–2080, Nov. 2012.

[95] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, Oct. 2009.

[96] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog Computing and Its Role in the Internet of Things," in *1st Workshop on Mobile Cloud Computing (MCC 2012)*, Helsinki, Finland: ACM, Aug. 2012, pp. 13–16.

[97] L. Codeca, R. Frank, and T. Engel, "Luxembourg SUMO Traffic (LuST) Scenario: 24 Hours of Mobility for Vehicular Networking Research," in *7th IEEE Vehicular Networking Conference (VNC 2015)*, Kyoto, Japan: IEEE, Dec. 2015.

[98] C. Sommer, R. German, and F. Dressler, "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis," *IEEE Transactions on Mobile Computing*, vol. 10, no. 1, pp. 3–15, Jan. 2011.

[99] A. Varga and R. Hornig, "An Overview of the OMNeT++ Simulation Environment," in *1st ACM/ICST International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools 2008)*, Marseille, France: ACM, Mar. 2008.

[100] K. Wessel, M. Swigulski, A. Köpke, and D. Willkomm, "MiXiM – The Physical Layer: An Architecture Overview," in *2nd ACM/ICST International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools 2009): 2nd ACM/ICST International Workshop on OMNeT++ (OMNeT++ 2009)*, Rome, Italy: ACM, Mar. 2009.

[101] C. Sommer, S. Joerer, and F. Dressler, "On the Applicability of Two-Ray Path Loss Models for Vehicular Network Simulation," in *4th IEEE Vehicular Networking Conference (VNC 2012)*, Seoul, Korea: IEEE, Nov. 2012, pp. 64–69.

[102] D. Eckhoff and C. Sommer, "A Multi-Channel IEEE 1609.4 and 802.11p EDCA Model for the Veins Framework," in *5th ACM/ICST International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools 2012): 5th ACM/ICST International Workshop on OMNeT++ (OMNeT++ 2012), Poster Session*, Desenzano, Italy: ACM, Mar. 2012.

[103] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent Development and Applications of SUMO - Simulation of Urban MObility," *IARIA International Journal On Advances in Systems and Measurements*, vol. 5, no. 3&4, pp. 128–138, Dec. 2012.

[104] A. Virdis, G. Stea, and G. Nardini, "SimuLTE - A Modular System-level Simulator for LTE/LTE-A Networks based on OMNeT++," in *4th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2014)*, Vienna, Austria, Aug. 2014.

[105] R. Riebl, H.-J. Günther, C. Facchi, and L. Wolf, "Artery - Extending Veins for VANET Applications," in *4th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS 2015)*, Budapest, Hungary: IEEE, Jun. 2015.

[106] A. Virdis, G. Nardini, and G. Stea, "Modeling unicast device-to-device communications with SimuLTE," in *1st International Workshop on Link- and System Level Simulations (IWSLS)*, Vienna, Austria: IEEE, Jul. 2016.

[107] S. Djahel, M. Salehie, I. Tal, and P. Jamshidi, "Adaptive Traffic Management for Secure and Efficient Emergency Services in Smart Cities," in *International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, San Diego, CA: IEEE, Mar. 2013, pp. 340–343.

[108] B. Havaei-Ahary, "Road Traffic Estimates: Great Britain 2017," Department for Transport, Report, Jul. 2018.

[109] O. Altintas, K. Seki, H. Kremo, M. Matsumoto, R. Onishi, and H. Tanaka, "Vehicles as Information Hubs During Disasters: Glueing Wi-Fi to TV White Space to Cellular Networks," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 1, pp. 68–71, Jan. 2014.

[110] "Powering Telecoms: East Africa Market Analysis," GSMA, Report, Oct. 2012.

[111] W. Viriyasitavat, O. K. Tonguz, and F. Bai, "Network Connectivity of VANETs in Urban Areas," in *6th IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON 2009)*, Rome, Italy: IEEE, Jun. 2009.

[112] J. Barros, "How to Build Vehicular Networks in the Real World," in *15th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2014)*, Philadelphia, PA: ACM, Aug. 2014, pp. 123–124.

[113] B. Baron, P. Spathis, H. Rivano, M. Dias De Amorim, Y. Viniotis, and J. Clarke, "Software-Defined Vehicular Backhaul," in *IFIP Wireless Days Conference 2014*, Rio de Janeiro, Brazil: IEEE, Nov. 2014.

[114] A. Mian, R. Baldoni, and R. Beraldi, "A Survey of Service Discovery Protocols in Multihop Mobile Ad Hoc Networks," *IEEE Pervasive Computing*, vol. 8, no. 1, pp. 66–74, Jan. 2009.

[115] F. Sailhan and V. Issarny, "Scalable Service Discovery for MANET," in *3rd IEEE International Conference on Pervasive Computing and Communications (PerCom 2005)*, Kauai, HI: IEEE, Mar. 2005, pp. 235–244.

[116] K. Abrougui, A. Boukerche, and R. Pazzi, "Design and Evaluation of Context-Aware and Location-Based Service Discovery Protocols for Vehicular Networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 3, pp. 717–735, Sep. 2011.

[117] A. Lakas, M. A. Serhani, and M. Boulmalf, "A Hybrid Cooperative Service Discovery Scheme for Mobile Services in VANET," in *7th International Conference onWireless and Mobile Computing, Networking and Communications (WiMob)*, Shanghai, China: IEEE, Oct. 2011, pp. 25–31.

[118] S. Noguchi, M. Tsukada, T. Ernst, A. Inomata, and K. Fujikawa, "Location-aware service discovery on IPv6 GeoNetworking for VANET," in *11th International Conference on ITS Telecommunications (ITST)*, St. Petersburg, Russia: IEEE, Aug. 2011, pp. 224–229.

[119] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A Survey of Information-Centric Networking," *IEEE Communications Magazine*, vol. 50, no. 7, pp. 26–36, Jul. 2012.

[120] M. Amadeo, C. Campolo, and A. Molinaro, "CRoWN: Content-Centric Networking in Vehicular Ad Hoc Networks," *IEEE Communications Letters*, vol. 16, no. 9, pp. 1380–1383, Sep. 2012.

[121] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, k. claffy kc, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named Data Networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 66–73, Jul. 2014.

[122] L. Wang, R. Wakikawa, R. Kuntz, R. Vuyyuru, and L. Zhang, "Data Naming in Vehicle-to-Vehicle Communications," in *31st IEEE Conference on Computer Communications (INFOCOM 2012): Workshop on Emerging Design Choices in Name-Oriented Networking*, Orlando, FL: IEEE, Mar. 2012, pp. 328–333.

[123] G. Grassi, D. Pesavento, L. Wang, G. Pau, R. Vuyyuru, R. Wakikawa, and L. Zhang, "ACM HotMobile 2013 Poster: Vehicular Inter-networking via Named Data," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 17, no. 3, pp. 23–24, Jul. 2013.

[124] N. B. Melazzi, A. Detti, M. Arumaithurai, and K. K. Ramakrishnan, "Internames: a name-to-name principle for the future Internet," in *10th International Heterogeneous Networking for Quality, Reliability, Security and Robustness (QShine 2014)*, Island of Rhodes, Greece: IEEE, Aug. 2014, pp. 146–151.

[125] S. Farrell, D. Kutscher, C. Dannewitz, B. Ohlman, A. Keranen, and P. Hallam-Baker, "Naming Things with Hashes," IETF, RFC 6920, Apr. 2013.

[126] A. Fonseca and T. Vazão, "Applicability of position-based routing for VANET in highways and urban environment," *Elsevier Journal of Network and Computer Applications*, vol. 36, no. 3, pp. 961–973, Mar. 2013.

[127] H. Füßler, H. Hartenstein, J. Widmer, and M. Mauve, "Contention-Based Forwarding for Street Scenarios," in *1st International Workshop on Intelligent Transportation (WIT)*, Hamburg, Germany, Mar. 2004, pp. 155–160.

[128] K. C. Lee, U. Lee, and M. Gerla, "Geo-Opportunistic Routing for Vehicular Networks," *IEEE Communications Magazine*, vol. 48, no. 5, pp. 164–170, May 2010.

[129] G. Al-Kubati, A. Al-Dubai, L. Mackenzie, and D. P. Pezaros, "Efficient Road Topology based Broadcast Protocol for VANETs," in *IEEE Wireless Communications and Networking Conference (WCNC 2014)*, Istanbul, Turkey: IEEE, Apr. 2014, pp. 2710–2715.

[130] P. Salvo, M. De Felice, F. Cuomo, and A. Baiocchi, "Infotainment traffic flow dissemination in an urban VANET," in *IEEE Global Telecommunications Conference (GLOBECOM 2012)*, Anaheim, CA: IEEE, Dec. 2012, pp. 67–72.

[131] ETSI, "Intelligent Transport Systems (ITS); Vehicular communications; GeoNetworking; Part 4: Geographical addressing and forwarding for point-to-point and point-to-multipoint communications; Sub-part 1: Media-Independent Functionality," ETSI, Tech. Rep. 302 636-4-1 V1.2.1, Jul. 2014.

[132] Japanese Ministry of Land Infrastructure and Transport; Road Bureau, "Nationwide Road and Street Traffic Conditions Survey - Vehicles Using Tomei Freeway (FY2010)," Japanese Ministry of Land, Infrastructure and Transport; Road Bureau, Report, Dec. 2012.

[133] Shizuoka Prefecture Planning Public Relations Department, "Tomei Expressway Interchange Traffic Volume (FY2010)," Shizuoka Prefecture Planning Public Relations Department, Report, May 2012.

[134] E. Lee, E.-K. Lee, M. Gerla, and S. Oh, "Vehicular Cloud Networking: Architecture and Design Principles," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 148–155, Feb. 2014.

[135] V. Sucasas, A. Radwan, H. Marques, J. Rodriguez, S. Vahid, and R. Tafazolli, "A survey on clustering techniques for cooperative wireless networks," *Elsevier Ad Hoc Networks*, vol. 47, pp. 53–81, Sep. 2016.

[136] M. Chatterjee, S. K. Das, and D. Turgut, "WCA : A Weighted Clustering Algorithm for Mobile," *Springer Journal of Cluster Computing*, vol. 5, no. 2, pp. 193–204, Apr. 2002.

[137] A. Daeinabi, A. G. Pour Rahbar, and A. Khademzadeh, "VWCA: An Efficient Clustering Algorithm in Vehicular Ad Hoc Networks," *ACM Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 207–222, Jan. 2011.

[138] H. Cheng, J. Cao, X. Wang, S. K. Das, and S. Yang, "Stability-aware multimetric clustering in mobile ad hoc networks with group mobility," *Wiley Wireless Communications and Mobile Computing (WCMC)*, vol. 9, no. 6, pp. 759–771, Jun. 2009.

[139] S. Z. H. Zahidi, F. Aloul, A. Sagahyroon, and W. El-Hajj, "Optimizing Complex Cluster Formation in MANETs Using SAT/ILP Techniques," *IEEE Sensors Journal*, vol. 13, no. 6, pp. 2400–2412, Jun. 2013.

[140] R. Atat, E. Yaacoub, M.-S. Alouini, and F. Filali, "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)*, Las Vegas, NV: IEEE, Jan. 2012, pp. 316–320.

[141] L. Bononi and M. Di Felice, "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, Oct. 2007.

[142] S. Khakpour, R. W. Pazzi, and K. El-Khatib, "Using clustering for target tracking in vehicular ad hoc networks," *Elsevier Vehicular Communications*, vol. 9, pp. 83–96, Jul. 2017.

[143] S.-S. Wang and Y.-S. Lin, "PassCAR: A passive clustering aided routing protocol for vehicular ad hoc networks," *Elsevier Computer Communications*, vol. 36, no. 2, pp. 170–179, Jan. 2013.

[144] S. Ucar, S. C. Ergen, and O. Ozkasap, "Multihop-Cluster-Based IEEE 802.11p and LTE Hybrid Architecture for VANET Safety Message Dissemination," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 4, pp. 2621–2636, Apr. 2016.

[145] N. Taherkhani and S. Pierre, "Centralized and Localized Data Congestion Control Strategy for Vehicular Ad Hoc Networks Using a Machine Learning Clustering Algorithm," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 11, pp. 3275–3285, Nov. 2016.

[146] M. M. Caballeros Morales, C. S. Hong, and Y.-C. Bang, "An Adaptable Mobility-Aware Clustering Algorithm in vehicular networks," in *13th Asia-Pacific Network Operations and Management Symposium (APNOMS 2011)*, Taipei, Taiwan: IEEE, Sep. 2011.

[147] K. C. Lee, M. Le, J. Härri, and M. Gerla, "LOUVRE: Landmark Overlays for Urban Vehicular Routing Environments," in *68th IEEE Vehicular Technology Conference (VTC2008-Fall)*, Calgary, Canada: IEEE, Sep. 2008, pp. 2157–2162.

[148] H. A. Omar, W. Zhuang, and L. Li, "VeMAC: A TDMA-Based MAC Protocol for Reliable Broadcast in VANETs," *IEEE Transactions on Mobile Computing*, vol. 12, no. 9, pp. 1724–1736, Sep. 2013.

[149] E. Dror, C. Avin, and Z. Lotker, "Fast randomized algorithm for 2-hops clustering in vehicular ad-hoc networks," *Elsevier Ad Hoc Networks*, vol. 11, no. 7, pp. 2002–2015, Sep. 2012.

[150] R. Crepaldi, M. Bakht, and R. Kravets, "QuickSilver: Application-driven Inter- and Intra-cluster Communication in Vanets," in *Third ACM International Workshop on Mobile Opportunistic Networks (MobiOpp '12)*, Zurich, Switzerland: ACM, Mar. 2012, 69–76.

[151] R. Goonewardene, F. Ali, and E. Stipidis, "Robust mobility adaptive clustering scheme with support for geographic routing for vehicular ad hoc networks," *IET Intelligent Transport Systems*, vol. 3, no. 2, pp. 148–158, Jun. 2009.

[152] S. Kumar, T. H. Lai, and A. Arora, "Barrier coverage with wireless sensors," *ACM/Springer Wireless Networks (WINET)*, vol. 13, no. 6, pp. 817–834, Dec. 2007.

[153] F. Dai and J. Wu, "Distributed Dominant Pruning in Ad Hoc Networks," in *IEEE International Conference on Communications (ICC 2013)*, Anchorage, AK: IEEE, May 2003, pp. 353–357.

[154] A. Ghosh, V. Vardhan, G. Mapp, O. Gemikonakli, and J. Loo, "Providing Ubiquitous Communication Using Road-Side Units in VANET Systems: Unveiling the Challenges," in *13th International Conference on ITS Telecommunications (ITST 2013)*, Tampere, Finland: IEEE, Nov. 2013, pp. 74–79.

[155] R. S. Bali, N. Kumar, and J. J. Rodrigues, "Clustering in vehicular ad hoc networks: Taxonomy, challenges and solutions," *Elsevier Vehicular Communications*, vol. 1, no. 3, pp. 134–152, Jul. 2014.

[156] A. Ghosh, V. Paranthaman, G. Mapp, O. Gemikonakli, and J. Loo, "Enabling Seamless V2I Communications: Toward Developing Cooperative Automotive Applications in VANET Systems," *IEEE Communications Magazine*, vol. 53, no. 12, pp. 80–86, Dec. 2015.

[157] C.-M. Huang, M.-S. Chiang, and T.-H. Hsu, "PFC: A packet forwarding control scheme for vehicle handover over the ITS networks," *Elsevier Computer Communications, Special Issue on Mobility Protocols for ITS/VANET*, vol. 31, no. 12, pp. 2815–2826, Jul. 2008.

[158] M. Mouton, G. Castignani, R. Frank, and T. Engel, "Enabling vehicular mobility in city-wide IEEE 802.11 networks through predictive handovers," *Elsevier Vehicular Communications*, vol. 2, no. 2, pp. 59–69, Apr. 2015.

[159] M. Raya, A. Aziz, and J.-P. Hubaux, "Efficient Secure Aggregation in VANETs," in *3rd International Workshop on Vehicular Ad Hoc Networks (VANET '06)*, Los Angeles, CA: ACM, Sep. 2006, 67–75.

[160] K. Ibrahim and M. C. Weigle, "CASCADE: Cluster-Based Accurate Syntactic Compression of Aggregated Data in VANETs," in *IEEE Global Telecommunications Conference (GLOBECOM 2008)*, New Orleans, LA: IEEE, Dec. 2008.

[161] A. Awad, R. German, and F. Dressler, "Exploiting Virtual Coordinates for Improved Routing Performance in Sensor Networks," *IEEE Transactions on Mobile Computing*, vol. 10, no. 9, pp. 1214–1226, Sep. 2011.

[162] J.-S. Park and S.-J. Oh, "A New Concave Hull Algorithm and Concaveness Measure for n-dimensional Datasets," *Academia Sinica Journal of Information Science and Engineering*, vol. 29, no. 2, pp. 587–600, Mar. 2013.

[163] S. Dietzel, J. Petit, F. Kargl, and B. Scheuermann, "In-Network Aggregation for Vehicular Ad Hoc Networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 1909–1932, Apr. 2014.

[164] F. Klingler, F. Dressler, and C. Sommer, "The Impact of Head of Line Blocking in Highly Dynamic WLANs," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 8, pp. 7664–7676, Aug. 2018.

# Acknowledgments

Foremost, I would like to thank my advisor Professor Falko Dressler for his guidance throughout my Ph.D. studies. His door was always open for discussions and his assistance helped me become a better scientist. I am also thankful for the assistance provided by the CCS-Team, especially by Agon, Christoph, Felix, and Florian. Working with you was always pleasant, informative, and often a lot of fun. Much of the work presented in this thesis was done in collaboration with Toyota ITC (both Japan and US). For the provided opportunities, among them a research visit to Tokyo, I would like to thank Dr. Onur Altintas and his team.

Furthermore, I want to thank my family, particularly my parents Ursula and Helge, and my sister Iris and her partner Clemens, for their support over the years. Finally, I would like to say thank you to my fiancée Melanie for her help and encouragement during my studies.

# Danksagung

An erster Stelle möchte ich mich bei meinem Betreuer, Professor Falko Dressler, für die Begleitung während des gesamten Promotionsstudiums bedanken. Seine Tür war immer offen für Diskussionen und seine Unterstützung half mir, ein besserer Wissenschaftler zu werden. Weiters bin ich dankbar für die Unterstützung durch das CCS-Team, insbesondere durch Agon, Christoph, Felix und Florian. Die Arbeit mit euch war immer angenehm, informativ und oft mit viel Spaß verbunden. Ein Großteil der in dieser Dissertation vorgestellten Arbeiten wurde in Zusammenarbeit mit Toyota ITC (Japan und USA) durchgeführt. Für die gebotenen Möglichkeiten, unter anderem einen Forschungsaufenthalt in Tokio, möchte ich mich bei Dr. Onur Altintas und seinem Team bedanken.

Darüber hinaus möchte ich meiner Familie, besonders meinen Eltern Ursula und Helge, sowie meiner Schwester Iris und ihrem Partner Clemens, für ihre Unterstützung im Laufe der Jahre danken. Abschließend möchte ich mich bei meiner Verlobten Melanie für ihre Hilfe und Ermutigung während meines Studiums bedanken.