

ulm university universität ulm

Fakultät für Ingenieurwissenschaften und Informatik
Institut für Verteilte Systeme

Cluster-based Aggregation for Inter-vehicle Communication

Master Thesis at Ulm University



Institut für Verteilte Systeme
Institute of Distributed Systems

Submitted by:
Mihail Balanici
mihail.balanici@uni-ulm.de

Examiners:
Prof. Dr. rer. nat. Frank Kargl
Prof. Dr.-Ing. Stefan Wesner

Supervisor:
Dipl.-Inf. Stefan Dietzel

2013

Fassung December 18, 2013

© 2013 Mihail Balanici

This work is licensed under the Creative Commons. Attribution-NonCommercial-ShareAlike 3.0 License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/3.0/de/ or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.

Satz: PDF-LaTeX 2$_\varepsilon$

Acknowledgments

First and foremost I would like to thank my supervisor, Mr. Stefan Dietzel, whose academic attainments and experience, as well as his patience and kindness have been invaluable to me. I am extremely thankful for the constantly updating food for thought he has been always providing me with, and the valuable suggestions and advices without which this master thesis would not have been possible. I have thoroughly enjoyed working under his guidance and I highly appreciate all his continuous support and help related to all aspects of my thesis work. And last but not least, I have to mention our fruitful cooperation and mutual work invested in writing a common conference paper, which has been a great experience for me.

I would also like to express my gratitude to Prof. Frank Kargl for the interesting and valuable discussions we had, related to my thesis and our paper, as well as for his comments and annotations that helped me a lot in improving both works.

I am also very thankful to Prof. Stefan Wesner for his consent and time taken to examine my work. I really appreciate that.

Last but not least, I am indebted to my classmates and friends, namely Nadim Iskandar and Eugeniu Semenciuc for their time and support in helping me with writing and enhancing my simulation code.

Most importantly, this thesis would not have been possible at all without the continuous moral support and care of my lovely family, who were constantly beside me, in spite of being far away.

Abstract

The present Master Thesis is focused on the design and evaluation of a cluster-based aggregation protocol, which defines a set of rules and procedures for data aggregation based on a cluster structure. The proposed protocol is regarded as a complex mechanism consisting of two component sub-protocols: a clustering algorithm, grouping vehicles into cluster entities, and an aggregation scheme, deploying in-network and hierarchical data aggregation atop the prebuilt clusters. A cluster-based aggregation, as a concept, introduces a novel approach to the VANET technology, since it is more specific to other research areas, such as Wireless Sensor Networks (WSNs). In WSNs clustering mechanisms have been successfully applied to solve both scalability and data quality issues. The core idea behind clustering is to divide the network into smaller groups of nodes (sensors) sharing similar locations, and elect a leader node, called cluster head, for each group. In this respect, clustering helps scalability, because nodes exchange and aggregate information within the cluster, and disseminate further only the merged results, thus saving bandwidth resources. Moreover, cluster heads can guarantee that calculations are duplicate-free and unbiased (i.e., all sensor reports are considered with equal weight) [1].

In vehicular networks however, the main challenge which prevents the application of WSN clustering protocols is the high mobility of vehicular nodes. For instance, if clusters are built using the same location-similarity-based logic, then vehicles driving on the similar road portion but with different velocities, will cause frequent changes of the cluster structure, fact which hinders the achievement of cluster stability. As a result, newer approaches specifically tailored for VANET environments start taking into account vehicle mobility, cluster head stability and driver's intents.

In the context of vehicular networks, among the advantages brought by organization of vehicles into clusters can be mentioned the more stable communication links and the longer lasting inter-vehicle contact time – two factors that have a major positive impact on the aggregation itself, and is lacking at non-clustering schemes. The cluster formation fundamental procedure was derived from the decision criteria found in aggregation mechanisms, that imply the comparison of two data structures for their similarity, before data fusion can be applied. In our clustering algorithm, clusters are built based on the dynamic characteristics displayed by nodes, and namely on the similarity of these dynamic parameters that characterizes the mobility with which nodes move relative to each other. In this regard, we introduce the new metric that best of all characterizes the mobility similarity, namely the *relative average velocity* between nodes. As a rule, clustering was always performed relying on distance and position coordinates of cars (location-based clustering), rather than solely on their dynamic characteristics (speed, acceleration/deceleration). By

changing this approach towards building clusters based exclusively on nodes' mobility similarity, we pursue the increasing of cluster stability and lifetime. The main advantage of the presented protocol is the fact that aggregation is concatenated and strongly linked with clustering, in the sense that common parameters (i.e. relative and average velocities) and procedures are concurrently used in both sub-protocols. Another important asset is that cluster-based aggregation could be also well suited to build various security mechanisms (e.g., interactive proofs, agreement protocols, authentication and privacy applications) on top of it. That is because security protocols often rely on interaction between nodes. In normal conditions this interaction becomes unstable due to the nodes' high mobility, however velocity-based clustering could solve this problem by introducing stable and reliable inter-vehicle connections.

Last but not least, we introduce in this work a novel and very interesting approach to hierarchical aggregation, which copes efficiently with data duplicates and multiple transmissions of similar data structures. This was particularly possible due to the application of Bloom filters, that have not been used for such purposes before. The introduction of a Time-To-Live parameter improves further the proposed aggregation scheme.

This Master Thesis is organized into seven chapters. The first chapter is devoted to the introduction of IVC technology, involving a short overview of the VANET properties and characteristics, as well as an insight into the data aggregation technique. Chapter two is intended for the analysis of requirements, based on which the present cluster-based aggregation protocol has been designed, while chapter three analyzes the related work and the previously developed protocols, that have some common features or resembling aspects with our protocol. Chapters four and five are dedicated to the protocol concept and design proposals, making a detailed explanation and analysis of the main features characterizing both clustering and aggregation sub-protocols. Chapter six involves a thorough evaluation of our clustering algorithm's functionality, performing a comparison with the existing location-based and velocity-based clustering mechanisms. An important attention is also paid to the analysis of our hierarchical aggregation scheme and its distinct features and properties. Finally, chapter eight concludes the thesis by drawing the main conclusions and comments related to the developed cluster-based aggregation protocol.

Contents

1. Introduction **1**

1.1. VANET Development 2

1.2. Network Architectures and Characteristics 3

1.3. VANET Applications 6

1.4. In-network and Hierarchical Aggregation in VANETs 8

2. Motivation Analysis **11**

3. Related Work **15**

3.1. Clustering Protocols 15

3.2. Aggregation Schemes 18

4. Protocol Concept **19**

5. Protocol Model **21**

5.1. Clustering Algorithm 21

5.1.1. Cluster Formation 22

5.1.2. Velocity Threshold Condition 24

5.1.3. Packet Structure 27

5.1.4. Cluster Maintenance 29

5.1.5. Re-clustering Mechanism 30

5.1.6. Inter-cluster Communication 32

5.2. Clustering Case Scenarios 34

5.3. Cluster-based Aggregation Scheme 35

5.3.1. In-network Aggregation 35

5.3.2. Hierarchical Aggregation 36

5.3.3. Challenges and Solutions 38

5.3.4. Duplicate Detection Algorithm (DDA) 40

5.3.5. Timing and TTL 42

5.4. Conclusions 44

6. Protocol Evaluation **47**
6.1. Clustering Algorithm Analysis 47
6.1.1. Inter-vehicle Contact Time 49
6.1.2. Cluster Stability 50
6.1.3. Cluster Membership Duration 55
6.2. Hierarchical Aggregation Scheme Evaluation 56
6.2.1. DDA Case Studies 57
6.2.2. Network Convergence 61
6.2.3. False Positives Phenomenon 62
6.2.4. Parameter Optimization 64
6.2.5. Aggregates Truncation 66
6.3. Conclusions 68

7. Conclusions and Future Work **69**

A. Appendix **71**

List of Figures **73**

List of Tables **75**

Bibliography **77**

Acronyms **81**

1

Introduction

Inter-vehicle Communication (IVC) is an actual emerging field of the ICT sector and a fast developing research area which has gotten an increased attention from researchers, engineers and car manufacturers from all around the world. This communication technique is mainly intended for improvement of driving safety and traffic efficiency on the road, thus implementing a large variety of applications and protocols. IVC finds its deployment in the so called Vehicular Ad-hoc Network (VANET) technology, which is a particular and similar in some aspects case of Mobile Ad-hoc Networks[1] (MANETs). Basically VANET is a highly dynamic network where cars/vehicles play the role of mobile-wireless nodes which communicate directly using ad-hoc connections, thereby mutually exchanging information via wireless broadcast.

The most important and major application of IVC in general and of VANETs in particular are the Intelligent Transport Systems (ITS), that include telematics, active road safety, traffic efficiency and management, infotainment, and all types of communication in vehicles, between vehicles (e.g. car-to-car), and between vehicles and fixed locations (e.g. car-to-infrastructure). However, ITS are not restricted to road transport only – they also include the use of information and communication technologies for rail, water and air transport facilities, including navigation systems (Figure 1.1) [3].

[1] A Mobile Ad-hoc Network (MANET) is an autonomous system made up of mobile stations interconnected by wireless connections without using certain management of a centralized infrastructure. Following existing communications in the network, the mobile nodes can also assume the role of routers to relay data [2].



Figure 1.1.: Intelligent Transport Systems [3].

1.1. VANET Development

In the past three decades, IVC and VANET research areas have drawn a significant attention from both academia and industry. Among the earliest studies on IVC are the one started by JSK (Association of Electronic Technology for Automobile Traffic and Driving) of Japan in the early 1980s, followed a bit later by such renowned projects as California PATH of US and Chauffeur of Europe [4]. Today, there are several tens of such famous research programs, which have extended throughout the world and acquired a continental, as well as an international recognition. In this context, industrial, governmental and academic research communities created significant opportunities and facilities for IVC study. US IntelliDrive(sm), CAMP/VSC-2, CICAS, Safe Trip21 and the earlier mentioned California PATH are just a few examples of such projects launched in the USA. The vehicular networking protocol standards deployed in such projects, except the SafeTrip21, are the WAVE (Wireless Access in Vehicular Environments) protocols suite, which was standardized by the IEEE under IEEE 802.11p (currently known as IEEE 802.11-2012) and IEEE 1609 protocol set. The SafeTrip21 project uses cellular technologies as wireless communication medium, instead of the one specified by IEEE 802.11-2012 [5].

In Europe and Japan, the standardization and development of ITS protocols and systems have also known a great success. Many projects have been initiated, whose main goals were oriented on different tasks, such as design, development and testing of technologies needed to support inter-vehicle and vehicle-to-infrastructure communications, design of

data security algorithms and communication protocols, development of geographic and logical addressing, as well as routing algorithms and different other solutions. Such famous European projects included CVIS, NoW, SEVECOM, COOPERS, GeoNET, PRECIOSA and many others. Among the Japanese counterparts can be mentioned: ASV, ETC, VICS, AHSRA and a few others [5].

One of the greatest contributions to the standardization of ITS on European level was brought by the European Telecommunications Standards Institute (ETSI) and Car2Car Communication Consortium (C2C-CC). The C2C-CC organization was initiated by the European vehicle manufacturers, that consequently attracted many research and academic associations for collaboration and partnership. The list of these institutions can be specified here as well, but this is out of the scope of this work.

1.2. Network Architectures and Characteristics

IVC systems can be split and classified according to their communication patterns into two main large categories:

1. **V2V Communications** (Vehicle-to-Vehicle Communications) which refer to the ad-hoc connections established between cars, and the inter-vehicle data transfer;
2. **V2I Communications** (Vehicle-to-Infrastructure Communications) characterizing the ad-hoc connections set between vehicles and the highway infrastructure consisting of intelligent wireless roadside units (RSUs).

Both types of VANETs are intended for exchange of critical safety and operational data, and are sometimes commonly referred to as **V2X Communications**.

According to Li *et al.* [4], IVC networks can be grouped depending on their architecture into three main classes: pure cellular/WLAN, pure ad-hoc and hybrid. In the first architecture scenario, VANET nodes make use of fixed cellular gateways or WLAN access points in order to gather traffic information, further disseminate their own data (by means of routing) or simply for establishing an infotainment connection to the Internet. In this way, VANETs combine both cellular network and WLAN capabilities to further extend their structure, where a wireless access point is used if available, or a UMTS/LTE connection is applied otherwise. Of course for such a scenario to become possible, a fixed gateway/access point infrastructure needs to be available, which eventually becomes unfeasible due to high costs involved. If so, a cheaper and more efficient solution would be the V2X connections, where all vehicles and roadside radio units form mobile ad-hoc networks in order to achieve the same goals as with a fixed communication infrastructure. Last but not least, the third, hybrid

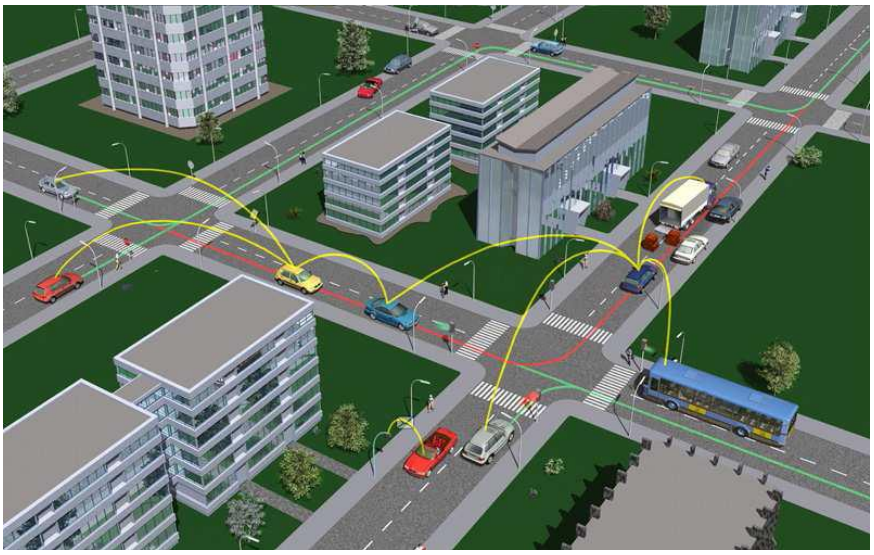


Figure 1.2.: Example of a VANET topology in a city-traffic scenario [6].

architecture scenario implies the combination of the cellular/WLAN and ad hoc networks, that could eventually lead to highly scalable and robust VANET structures. In the present work we are mainly focusing on inter-vehicle communication networks, implying the V2V ad-hoc topologies.

As discussed, VANETs are a special type of MANETs, where certain features such as nodes' mobility, network self-organization and self-management, a relatively short transmission range and a low bandwidth are common for both technologies. Nevertheless, there is a set of characteristics that distinguishes VANETs from other ad-hoc networks, thus making them unique. Among such featuring aspects are:

1. **Moving patterns.** Different from MANETs, which can contain a large number of nodes moving chaotically (i.e. according to an uncontrolled moving pattern), VANET nodes move following a strict motion path determined by factors like road trajectory, number of lanes, traffic density and traffic regulations. Because of the predefined node movement, it is easier to provide VANETs with some supporting fixed infrastructure services, such as the earlier mentioned WLAN access points, or RSUs for critical or dangerous locations like busy intersections, slippery roads, etc. Moreover, the preestablished moving patterns permit to model the node's mobility and predict with a certain probability the future node's position.

2. **Highly dynamic topology.** Due to high velocities of moving vehicles (up to 250 km/h), the VANET topology is constantly changing. This phenomenon hampers a stable communication and data exchange process, leading to a highly variable number of communication partners and frequent nodes' disconnections from the network. This is particularly challenging when such infotainment applications as

connection to the Internet or a file transfer application are provided through inter-vehicle communications.

3. **Sufficient energy and storage resources.** As compared to MANETs, VANET nodes possess a large amount of energy and computing power (including both storage and processing resources), since nodes are cars instead of tiny handheld devices [4].

4. **Communication environments.** VANETs usually operate in different communication environments, starting with low-density rural regions, and ending up with highly dense metropolitan and city areas. These environments influence the VANET functionality from two main perspectives:

   a) *Traffic density.* In rural and small towns, the traffic density proves to be much lower compared to cities and highly populated regions. This fact imposes additional demands for VANET connectivity support, which can be solved using fixed infrastructure communication facilities (RSUs, WLAN APs, etc).

   b) *Radio-signal propagation.* Rural areas and highways offer the advantage of straightforward and idle paths for signal propagation, fact which eases the connection establishment and ensures more durable inter-vehicle communication links. On the other hand, in city scenarios such communication-affecting phenomena as multipath fading, shadowing and Doppler frequency shifts caused by buildings, trees and other obstacles have to be taken into account and compensated as much as possible.

5. **High delay constraints.** In vehicular ad-hoc networks, the delay of data transmission is a crucial parameter having a tremendous impact on the driving security and efficiency. Such applications as emergency brake lights, blind spot and lane change warnings, pre-crash sensing and many others need a low dissemination delay so that all the adjacent nodes are immediately informed. These constraints can be applied using special QoS and message prioritization policies.

6. **Interaction with onboard equipment.** It is well known that modern vehicles are equipped with a series of onboard instruments, which offer security, driver assistance and infotainment services. This instrumentation includes different types of sensors, radar and radio communication facilities, Global Positioning Systems, frontal, lateral and rear view cameras, data collection and processing equipment, while the newest models of Mercedes-Benz cars are also equipped with a special type of road scanning devices (3D-cameras), and many others (Figure 1.3). In this context, all onboard instruments help to acquire precise data about the environment, road and weather

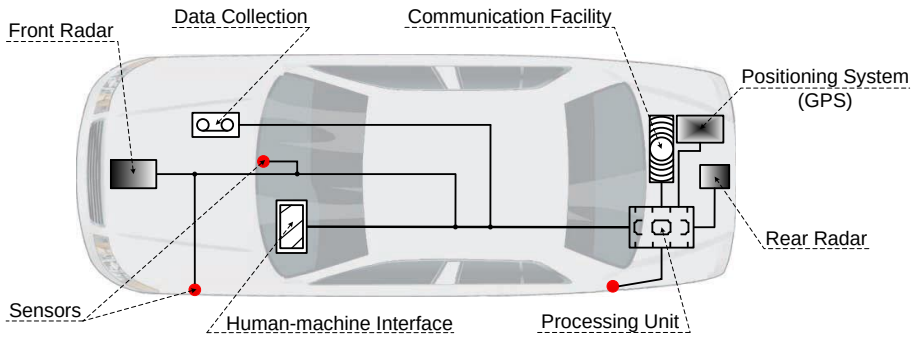


Figure 1.3.: A VANET node's onboard equipment (based on [7]).

conditions, traffic situation, etc, which subsequently can be easily shared among moving vehicles, thus making driving more secure, more comfortable and efficient.

1.3. VANET Applications

Inter-vehicle Communications offer a wide range of applications, varying from simple exchange of vehicle status information, to sophisticated and highly complex traffic systems. VANETs can successfully be deployed in active safety driving systems, where messages about accidents, road surface and weather conditions, approaching of an emergency vehicle, and other critical or hazardous situations are announced hundreds of meters or kilometers beforehand. These caution methods ensure a much safer driving, since drivers know what to expect in an area that is out of their line-of-sight, fact which subsequently permits them to take further actions. Among other important applications that VANETs offer are the enhanced and comfort driving, traffic condition monitoring and dynamic route scheduling. IVC systems introduce the dissemination of traffic jam-related data, intended to give drivers the opportunity of taking another path in order to avoid a certain congested road segment. Last but not least, among other applications can also be found the dissemination of the number of free parking places within a parking lot area, or the announcement of restaurants, hotels or gas stations along the highway.

As it can be seen, the range of services offered by VANETs is very large and diverse, therefore a classification of these applications is very useful for further analysis. A very good systematization of ITS services is presented by Schoch *et al.* in [8] (Table 1.1). According to this organization, there are four distinct groups of applications, including active safety, public service, improved driving and business/entertainment. **Active safety** applications are considered as the most necessary and important group of services offered by VANETs, since they have a direct impact on road safety. As mentioned, the basic aim is to make driving safer by means of communication, where drivers are warned

about hazardous traffic situations or assist them in reacting appropriately if the accident cannot be avoided anymore [8]. The **public service** group of applications is intended for

Table 1.1.: Types of VANET Applications [8].

Application type	Situation/purpose	Application examples
I. Active safety	1. Dangerous road features	1. Curve speed warning, 2. Low bridge warning, 3. Warning about violated traffic lights or stop signals;
	2. Abnormal traffic and road conditions	1. Vehicle-based road condition warning, 2. Infrastructure-based road condition warning, 3. Visibility enhancer, 4. Work zone warning;
	3. Danger of collision	1. Blind spot warning, 2. Lane change warning, 3. Intersection collision warning, 4. Forward/rear collision warning, 5. Emergency electronic brake lights, 6. Rail collision warning, 7. Warning about pedestrians crossing;
	4. Crash imminent	1. Pre-crash sensing;
	5. Incident occurred	1. Post-crash warning, 2. Breakdown warning, 3. SOS service.
II. Public service	1. Emergency response	1. Approaching emergency vehicle warning, 2. Emergency vehicle signal preemption, 3. Emergency vehicle at scene warning;
	2. Support for authorities	1. Electronic license plate, 2. Electronic drivers license, 3. Vehicle safety inspection, 4. Stolen vehicles tracking.
III. Improved driving	1. Enhanced Driving	1. Highway merge assistant, 2. Left turn assistant, 3. Cooperative adaptive cruise control, 4. Cooperative glare reduction, 5. In-vehicle signage, 6. Adaptive drive-train management;
	2. Traffic Efficiency	1. Notification of crash or road surface conditions to a traffic operation center, 2. Intelligent traffic flow control, 3. Enhanced route guidance and navigation, 4. Map download/update, 5. Parking spot locator service.
IV. Business/ Entertainment	1. Vehicle Maintenance	1. Wireless diagnostics, 2. Software update/flashing, 3. Safety recall notice, 4. Just-in-time repair notification;
	2. Mobile Services	1. Internet service provisioning, 2. Instant messaging, 3. Point-of-interest notification;
	3. Enterprise solutions	1. Fleet management, 2. Rental car processing, 3. Area access control, 4. Hazardous material cargo tracking;
	4. E-Payment	1. Toll collection, 2. Parking payment, 3. Gas payment.

supporting emergency vehicles and official motorcades, thus notifying regular drivers in advance and permitting them to create the necessary traffic corridors. However, this type

of applications has to be highly secured and protected, so that no intruding parties can abuse of them. The **improved driving** category of services includes applications intended to simplify or enhance driving by implementing driver assistance systems, thus making it more efficient and optimal. The case with traffic jam warning is a perfect example in this context. Finally, the **business/entertainment** group offers infotainment applications to drivers and passengers, thereby making driving more comfortable.

1.4. In-network and Hierarchical Aggregation in VANETs

Characteristic for vehicular networks is the ephemeral and unstable nature of data communication due to high vehicle mobility. Moreover, information is usually not addressed at specific receivers, but is broadcasted to a set of vehicles, and is often intended for large dissemination areas. The high mobility and active dynamics specific to VANET nodes, as well as the characteristics of radio propagation impose strict technical demands on the medium access schemes, bandwidth sharing mechanisms, supported data traffics and QoS requirements. In this context, clustering protocols offer an efficient solution for mitigation of these constraints, by simply providing stable and long-lasting inter-vehicle connections. Nevertheless, one common issue of IVC remains being the limited bandwidth of the wireless medium. With an increased traffic density, such as rush hours and traffic jams, VANET nodes compete for the medium access. This process can subsequently cause a running out of bandwidth resources, if every vehicle starts disseminating its individual data. In this respect, the necessity of an efficient radio resource management becomes obvious.

One solution for coping with this problem is the data compression mechanism, where before accessing the medium and performing packet transmission, a node compresses its data in order to occupy as little bandwidth as possible. This method however has some technical constraints – the data can only be compressed within certain limits in order to be able to properly decompress it at reception. Therefore, a slight decrease in the amount of disseminated information per node does not result in the highest efficiency.

Another solution which addresses this problem is the data aggregation method. It can be compared to a lossy-compression technique, because at some point information accuracy gets affected to the advantage of a lower amount of data transmitted over the medium. The basic idea behind aggregation is that correlated data and information with a high degree of similarity can be merged into a new data structure (Figure 1.4). This is due to the fact that many applications do not require atomic information, such as the precise position or velocity values characterizing vehicles. Instead, an abstract, aggregated view of the traffic situation on the road is pretty much sufficient [9]. A very good example is

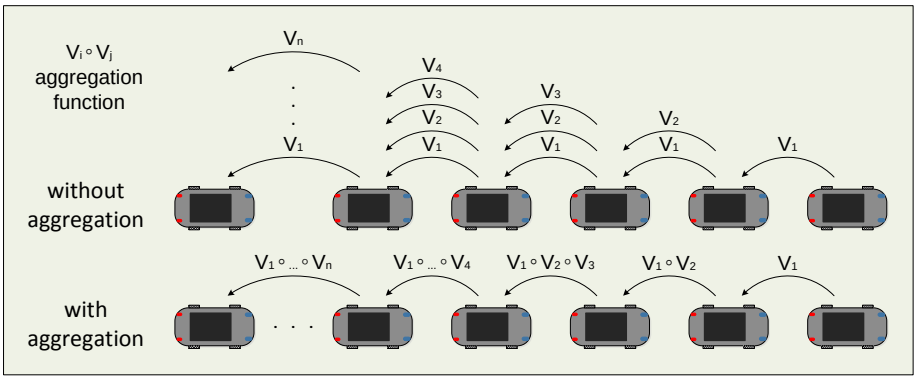


Figure 1.4.: Aggregation vs. Non-aggregation Schemes (based on [10]).

the earlier mentioned traffic jam or rush hour scenario, when nodes have almost similar velocities (which differ insignificantly). Therefore, all these speeds can be aggregated into a new value, which will characterize the average velocity of the jam structure. By applying this rule, the resulting single value is disseminated to other nodes approaching the queue, instead of broadcasting tens or even hundreds of individual data. As it is implicitly clear, this technique saves a lot of bandwidth and makes communication much more efficient.

Every aggregation scheme shares a series of common features and its functionality is based on several basic components. Dietzel *et al.* [10] systematize these components into four distinct elements (Figure 1.5):

1. **World Model.** The world model can be understood as a data base which accumulates all the information available to a certain node, be it self generated data (using the onboard instrumentation, e.g. sensors and cameras), or foreign data received from adjacent nodes. In this respect, the information contained by the world model is continuously changing and updating, as long as the node keeps observing and listening to the environment. If we refer to Figure 1.3 presented earlier, the world model component is part of the Data Collection module [10].

2. **Decision Criteria.** Any aggregation system has first to decide whether two or more data items are similar enough to be aggregated. To reach the decision, all information available to a node (current self-generated or contained in the world model) can be used. The decision component checks the world model periodically in order to discover similar or resembling data items that could be potentially aggregated. The data similarity is compared against a certain similarity-degree threshold, which basically defines the accuracy of the data aggregation scheme (or how lossy the aggregation system is). Thus, one of the main requirements for the decision criteria is *flexibility*. This means the decision component has to be able to combine repeating information on a long road segment, such as velocities within a traffic jam, while

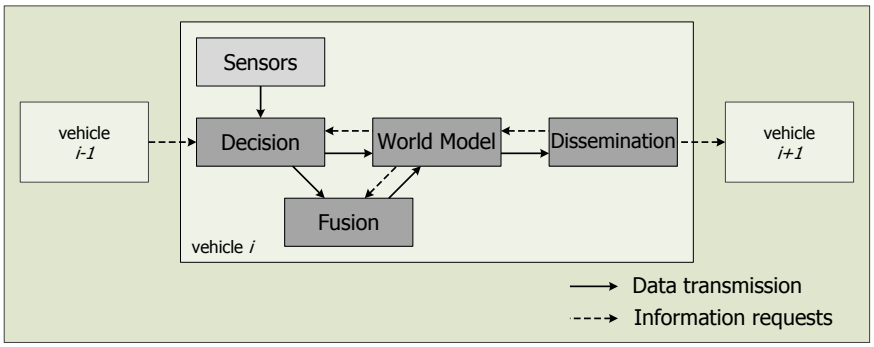


Figure 1.5.: Architectural model of an aggregation scheme [10].

being able at the same time to not overlook small details, differing from the general picture of the traffic situation (e.g. a car crash) [9], [10].

3. **Information Fusion.** Once the decision component has decided that two or more data items are similar enough to be aggregated, the information fusion procedure comes into action. There are several fusion functions, applicable in the context of data aggregation. Among them are averaging of two data values, taking the maximum or minimum value, counting of data items, logical operations, and others. For the fusion scheme to function properly, application-specific knowledge is necessary. Such information is very helpful when it comes to deciding which data items are more important for a specific application, and which of them can be discarded without losing accuracy or valuable details [9], [10].

4. **Dissemination Mechanism.** After the aggregation of fusible data items, the newly formed aggregate needs to be disseminated in the surrounding area, in order to inform the adjacent nodes about the world model of the source vehicle. Basically, if the amount of information contained in the world model fits into the bandwidth limits preestablished for one data transmission (the so-called *bandwidth profile*), then the entire world model can be disseminated. Otherwise, only the most relevant data (for instance, based on the application type) is selected for further dissemination. Flooding/broadcasting, periodic beaconing and geo-broadcasting are a few examples of data dissemination methods [10].

Hierarchical aggregation is a particular example of in-network aggregation, where instead of individual data items, available similar aggregates are further aggregated. The main goal pursued by hierarchical aggregation is the reducing of data granularity with increasing distance. Without this functionality, aggregation schemes cannot scale to large dissemination ares [11]. Hierarchical aggregation will be dealt with in more detail in the upcoming Chapters 5 and 6.

2

Motivation Analysis

As discussed in the first chapter, one of the greatest problems characteristic to VANETs is the limited bandwidth of the wireless communication medium. An increased traffic density leads to an aggravation of this problem, since every node starts competing for the medium access. Thus, the nodes' density has a direct impact on the success rate of inter-vehicle data transmissions, collision rate, and communication efficiency as a whole. As a result, taking corresponding measures necessary to solve this issue and subsequently improve the radio resource management becomes utterly important. A good solution in addressing the bandwidth shortage problem is the data aggregation method, narrowly analyzed in the previous chapter.

Another important aspect and deficiency of IVC systems is the maximum distance at which data dissemination can be performed. Information about critical traffic situations (e.g. congestions or car accidents) should be broadcasted within a few kilometers range in order to give the approaching drivers enough time for decision taking and a proper acting. In single-hop scenarios, packet transmission is an easy manageable task, however difficulties arise for multi-hop, long distance disseminations. In this respect, clustering mechanisms can be a very helpful solution in overcoming this problem. That is, nodes organize themselves into cluster structures, that will normally span a much larger sensed area than a single vehicle would cover. This gives an extended vision of the ahead upcoming traffic situations to vehicles driving behind, especially to those forming the

rear edge of the cluster. On the other hand, inter-cluster communication will permit the extension of this view even further, permitting the behind driving clusters to get informations about upcoming traffic scenarios in a range which is out of the cluster's communication radius. In such conditions, the bandwidth resources can again get affected, since too many clusters would disseminate their individual data. Therefore, clustering combined with hierarchical aggregation should be a very good and promising solution for coping with these limitations. By reducing data granularity using the hierarchical aggregation of in-network aggregates, the data aggregation scheme will be able to scale well to large dissemination ares. Important is to emphasize that the in-network aggregation could be successfully organized atop the cluster structure – procedure which leads to our designed cluster-based aggregation scheme, and in fact the research topic of this work.

The cluster-based in-network aggregation protocol developed in this master thesis is intended for aggregation of multiple dynamic characteristics, namely the node average velocities displayed by vehicles participating in cluster formation. The aggregation function used with this purpose is the classical averaging of nodes' velocities. As a result, the final aggregate value characterizes the cluster average velocity. The resulting data aggregates are further hierarchically aggregated if two or more clusters enter in mutual contact. The main aim is to drastically reduce the amount of velocity-related data transmitted via wireless medium in order to save radio resources and ensure a higher reliability of inter-vehicle and inter-cluster communications. In this context, clustering comes as an improvement technique for aggregation modernization – nodes are grouped into clusters, thus aggregation becomes centralized, cluster-based.

One of the main motivations which led to the design and analysis of this protocol is its novelty and the relatively unknown behavior of cluster-based aggregation in VANET environments. The main uncertainty comes from the fact that cluster-based aggregation is more specific to Wireless Sensor Networks (WSNs), where such schemes as LEACH [12] and COUGAR [13] have been successfully developed. In IVC systems this aggregation technique is less common because of the active dynamics of VANET nodes and the relatively problematic organization of vehicles into clusters. Primarily due to this reason, we organize clustering by taking into account the mobility patterns displayed by nodes, thus establishing stable and long-termed cluster entities. By considering the similarity degree of these dynamic features, clusters are built exclusively among vehicles sharing similar or resembling mobility behaviors. With that said, in our approach, a cluster is considered the basic entity atop which in-network aggregation is performed.

An important aspect that needed to be taken into account while developing our protocol was the consideration of present ITS standards and regulations. One of the main

reasons for this consideration is the fact that modern simulators, used in protocol analysis, work based on standardized parameters and specifications. According to the ETSI Standardization documentation for Intelligent Transport Systems, two important IVC services were chosen as baseline for our protocol development. This design approach permits the integration of these services in our protocol, thus playing ultimately an important supporting role in its functionality. The two services are called Local Dynamic Map (LDM) and Cooperative Awareness Message (CAM). In co-operative Intelligent Transport Systems, the Local Dynamic Map is a key facility element that supports various ITS applications by maintaining the information on different objects influencing or being part of traffic. ITS applications are designed to process static, temporary and dynamic data from other ITS nodes located in the surrounding area of the host station (be it a vehicle or a roadside unit). Hence, the relevant data needs to be stored and maintained in the LDM. The information contained in the LDM is received from relevant messages such as ITS CAM (Cooperative Awareness Messages), DENM (Decentralized Environmental Notification Messages), etc. LDM is a conceptual data store that is embedded in an ITS station and which maintains topographical, positional and status information related to the surrounding ITS stations. It is associated with the Data Collection module of the onboard equipment (Figure 1.3) and with the World Model component of aggregation protocols analyzed earlier. This means that common information required by different applications can be maintained in the LDM. Data is basically received from a range of different sources such as vehicles, infrastructure units, traffic centers and on-board sensors [14].

The second service of interest related to our protocol and one of the data types describing real world objects is the Cooperative Awareness Message (CAM) which refers namely to highly dynamic data. It includes information about ITS nodes located in close proximity, such as vehicles and dynamic traffic signs. It also describes information of the real world with a highly dynamic behavior, having mainly influence on traffic safety and efficiency (e.g. current speed, position and heading direction of adjacent stations moving within the communication range of the host, etc). CAMs are distributed within the ITS-G5 (802.11-2012) network and provide information of presence, position, as well as basic status of communicating ITS nodes to neighboring ITS stations that are located within a single hop distance. All nodes have to be able to generate, send and receive CAMs, as long as they participate in V2X communications. By receiving CAMs, the ITS station is aware of other stations in its neighborhood area as well as their positions, movement, basic attributes and basic sensor data. At the receiver side, reasonable efforts can be taken to evaluate the relevance of the messages and their information. This allows an ITS station to get acquainted about its situation and act accordingly. Observing several rules, the CAM Management entity generates CAMs using such facilities as time management,

station state monitoring and/or mobile station dynamic as data sources. Subsequently, valid CAMs are passed to the LDM management, which analyses the messages and updates in real time the LDM data base. As it has been mentioned, the LDM data base is a local geo-referenced database containing a C2X-relevant image of the real world. Applications retrieve these data by means of the LDM Management [15]. CAMs can be generated with variable intervals/frequencies, with the maximum value corresponding to: $T_{gen(max)} = 1\,s$, (i.e., $f_{gen(min)} = 1\,Hz$), and minimum – $T_{gen(min)} = 0,1\,s$, (i.e., $f_{gen(max)} = 10\,Hz$).

Another standardization aspect atop which we build our protocol is the Dedicated Short Range Communication (DSRC) technology, featuring such technical characteristics as:

1. short and medium range wireless communications, with a data rate of 6-11 Mbps;
2. 30 MHz radio frequency band (European standard according to ETSI) at a 5.9 GHz carrier frequency;
3. frequency band divided into 7 channels: 1 control channel (used for exchanging security messages and advertisements) and 6 service channels (used for different kind of applications like road/weather condition messages, number of available parking places, IM, etc);
4. merged into IEEE 802.11-2012, formerly standardized under IEEE 802.11p;
5. transmission range within 250 m – 1000 m (in our particular case we consider $R_c = 250\,m$).

The main motivation behind the design and evaluation of our cluster-based aggregation protocol has been the tendency towards organization and establishment of stable and durable inter-vehicle communication links, atop which in-network and hierarchical aggregation could be organized. This is particularly beneficial for in-network data aggregation schemes, since clustering ensures that only nodes with similar or resembling dynamic features will participate in cluster formation – fact which facilitates the subsequent aggregation deployment. Another important goal pursued in this work was to support the hierarchical data aggregation by means of clustering. It will be shown later that namely the Cluster ID parameter, characterizing each cluster structure, also plays a crucial role in aggregates' identification and their consequent hierarchical aggregation. This is a novel approach, where both clustering and aggregation mechanisms benefit from each other. That is, clustering applies the decision criteria logic found in aggregation schemes, for clusters' organization, while aggregation gains from using the more stable connections among nodes (in-network aggregation), and ultimately uses the Cluster IDs for reliable and duplicate-insensitive data fusion (hierarchical aggregation).

3

Related Work

The cluster-based aggregation protocol designed and evaluated in this master thesis has a complex structure consisting of two sub-protocols: a clustering algorithm and a data-aggregation scheme. In this respect, the main theoretical background of this work is based on a number of literature items related to these two mechanisms. The following analysis presents an overview of different papers primarily focusing on clustering and aggregation schemes deployed in Vehicular Ad-hoc Networks.

3.1. Clustering Protocols

Talking about clustering algorithms, as a rule they are part of more complex protocols whose final purpose does not reduce to cluster formation only, instead includes such functionalities as routing of data packets or medium access schemes. In this context, the pre-established clusters play the role of entity structures atop which protocols operate. Santos *et al.* [16] present the so-called Cluster-Based Location Routing (CBLR) protocol intended for routing of data packets in VANETs. CBLR enables dynamic, reactive and self-organizing cluster structures among moving vehicles. It is based on and is very similar to Cluster-Based Routing Protocol (CBRP) [17] primarily designed for MANET environments. The main aspect of CBLR is that vehicles (also called nodes) organize

themselves into clusters, where a specific node plays the role of the Cluster Head (CH), while other participating nodes are regarded as Cluster Members (M). Before joining a cluster, each node passes through a special state called Undecided (UN), during which it starts a timer and broadcasts a beacon called Hello Message (HM). If the timer expires before a reply is received, this node promotes itself to a Cluster Head. Otherwise, if the Undecided node gets a reply from a Cluster Head, it becomes a Cluster Member [16]. An important feature of CBLR is characterized by the Bridge nodes, which are represented by vehicles sensing two or more Cluster Heads in their transmission range. These nodes are specifically used by Cluster Heads for inter-cluster communication and routing purposes. Both CBLR and CBRP base their clustering decisions solely on location closeness and do not take into account the vehicle's mobility characteristics, fact which can lead to often changing cluster memberships (i.e. cluster instability). Important is to emphasize that our clustering sub-protocol is an adaptation of CBLR, where some of its functionality aspects (e.g. the roles played by nodes) have been preserved and a few novel features have been introduced.

Beside CBLR, there are several other cluster-based routing protocols particularly designed to fit the requirements imposed by Inter-vehicle Communications. The reason why CBLR was chosen to the detriment of other analogue protocols is because the latter do not satisfy certain features required by our clustering algorithm. Thus, in Cluster-Based Directional Routing Protocol (CBDRP) [18] nodes driving in the same direction are grouped into clusters, where a cluster head is elected. The main drawback preventing CBDRP application is that its intra-cluster communication is based on a unicast transmission between CMs and CH, while inter-cluster communication is performed only by cluster heads who have to mutually sense each other. As a consequence, this protocol does not satisfy the requirements of our clustering algorithm, where packet forwarding is based on single-hop broadcast (beaconing), and inter-cluster communication is organized by one or multiple cluster members called Gateways. Also, a distinct feature preventing the CBDRP deployment is its location-based clustering, i.e., only nodes entering the mutual transmission range are clustered, regardless of their mobility characteristics. The Cluster Based Routing (CBR) Protocol [19], which is another location-based clustering mechanism, does not satisfy the most important requirement set by our algorithm — the clustering metric. In CBR the geographic location information is considered for clustering and routing purposes, and no dynamic characteristics are evaluated.

Günter *et al.* [20] propose a cluster-based medium access control (CBMAC) scheme, using a clustering mechanism adapted and based on CBLR. Basically the cluster formation algorithm is quite alike to its CBLR counterpart, the nodes playing similar roles in cluster establishment (undecided, member, gateway and cluster head). The authors introduce

small amendments to CBLR's functionality, as well as propose a special re-clustering procedure, when two or more clusters have to merge due to mutually sensing each other Cluster Heads. The decision which node leaves the CH state is based on a weighting factor that takes into account three parameters: the mobility, the node connectivity, and the mean distance to the neighbors. The contest is won by the CH with the lowest weighting factor.

Another interesting approach in clustering was introduced by Souza *et al.* [21], the so called Aggregate Local Mobility (ALM) Clustering Algorithm. This protocol is another example of CBLR adaptation and has many common features with CBMAC. The authors argue that CBMAC may suffer from rapidly changing and unstable CHs, therefore they propose an improvement to that drawback. The main difference between the two consists in the re-clustering organization when two CHs encounter each other. In the ALM algorithm, re-clustering is delayed in order to prevent over-eager re-organization of clusters upon an accidental contact of two CHs. Moreover, when re-clustering becomes inevitable, the ALM weight is considered in decision taking regarding who preserves its Cluster Head role, and who becomes a Member of the new merged cluster. The ALM parameter is calculated based on the ratio of two successive takes of the distance between a node and its neighbor in order to determine the relative mobility within the cluster. The CH with a lower ALM value maintains its state.

Rawshdeh *et al.* [22] use a new clustering concept that is completely different from the one used in CBLR. In their approach, the node's velocity serves as metric in cluster formation. The slowest node within a group of vehicles moving in mutual communication range initiates the cluster formation procedure, which consequently triggers all members with a higher velocity set their temporal cluster ID similar to the ID of the Cluster Originating Node (COV). These vehicles then start calculating their eligibility to become a CH based on their relative velocity as compared with other sensed nodes. A node announces its CH suitability only if it is greater than the previously received ones notified by other vehicles [22]. The winner node declares its ID to all other members. Further cluster joining and leaving are again decided using the relative velocity to the cluster head, which has to be less or equal to a predefined threshold value. The main shortage of this approach is that the algorithm does not consider the average speed of the cluster structure as a whole for joining/leaving decisions, but rather takes into account the velocity of the cluster head. This can lead to the presence of member nodes whose mutual relative velocity exceeds the threshold, despite their relative speeds to the cluster head are in the admissible range. Moreover, the authors imply that only the current speed of the cluster head is considered and not its development over time, which is important in filtering short term driving maneuvers, such as short overtaking accelerations.

3.2. Aggregation Schemes

As far as the data aggregation scheme is concerned, the following analysis presents a short overview of several algorithms comparable with our designed sub-protocol. Wischhof *et al.* [23] introduce the Self-organizing Traffic Information System (SOTIS), which is primarily a data abstraction and dissemination protocol. In the data abstraction process, a data aggregation function (based on a road segmentation metric, combined with a time stamp) is applied. As a result, the aggregated data is represented by the average velocity on a particular road segment. A drawback specific to SOTIS is that the tuples with the newest time stamps do not necessarily contain the most up-to-dated information, since time stamps are assigned when the aggregates are computed. Moreover, aggregation schemes based on fixed road segment lengths do not scale well in conditions of high traffic density or large dissemination areas.

Another interesting aggregation and dissemination mechanism proposed by Nadeem *et al.* [24] is TrafficView. Compared to SOTIS, TrafficView performs aggregation based on node's speed and position characteristics. Therefore, vehicles moving in mutual proximity are aggregated by calculating the averages of these two parameters and adding time stamps to provide additional informations about up-to-dateness of the aggregated data.

A different aggregation scheme based on the fuzzy reasoning has been proposed by Dietzel *et al.* [9]. According to their algorithm, all factors that influence the aggregation decision are expressed using the Fuzzy Set Theory. Then fuzzy logic operations are applied to reason about the influences, and finally, a decision is taken. The fuzzy logic based approach uses the dynamic (speed) characteristics of moving nodes when deciding whether data can be aggregated, which is a feature similar to our developed protocol.

Ibrahim and Weigle [25] present a cluster-based aggregation scheme suitable for dissemination of vehicle velocities. Different from SOTIS and TrafficView, the so-called CASCADE system employs syntactic and lossless data compression. The local view is clustered using fixed size segments and the resulting data tuples are further compressed using differential coding. Subsequently, the compressed information is disseminated using geo-broadcast.

From this analysis it should be concluded that CBLR's clustering algorithm is best suited for our purposes, therefore, we start with the same cluster messaging patterns. As far as the aggregation schemes are concerned, their main drawbacks are related to the use of fixed road segments for aggregation decisions, which results in scalability issues, as well as the application of generic timestamps, that possess a relativistic nature (i.e., necessity of timestamps comparison). We introduce a few mechanisms for addressing of these issues.

4

Protocol Concept

When designing our Cluster-based Aggregation Protocol (further on referred to as CBAP), we kept in mind two main goals. First of all, we wanted to propose an aggregation scheme that could be entirely based on and even concatenated with a clustering mechanism. Secondly, we tried to propose a clustering algorithm making use of Decision Criteria found in aggregation schemes, while focusing our attention on dynamic parameters characterizing nodes, rather than using their location characteristics. In this respect, nodes decide to organize themselves or not into common cluster structures by simply considering/analyzing the similarity degree of their dynamic characteristics – an identical procedure used in some decision criteria mechanisms. We believe that such a clustering approach will permit the organization of aggregation atop cluster structures, where data fusion will be carried out within the pre-established clusters.

The concept of aggregation applied atop clusters is particularly beneficial for the in-network data aggregation scheme, since clustering ensures that only nodes with similar or resembling dynamic features will participate in cluster formation – fact which facilitates the subsequent data aggregation procedure. Another important advantage is that clustering is also intended to support the hierarchical data aggregation. In-network aggregates are associated with a special parameter, called Cluster ID, making each cluster structure unique. This parameter plays a crucial role in aggregates' identification and their consequent hierarchical aggregation. This is a novel approach, where both clustering and aggregation

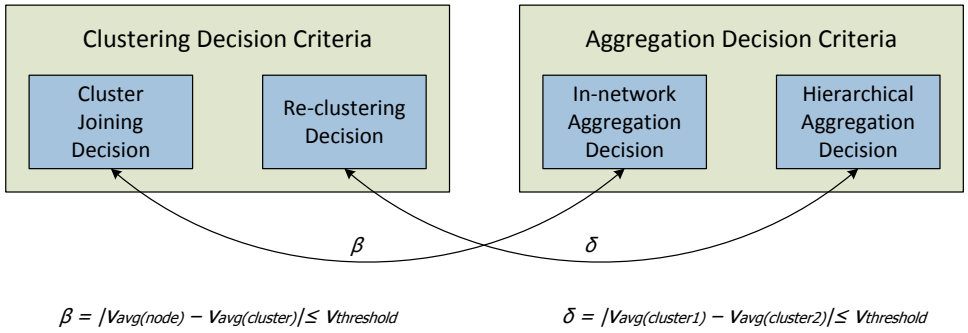


Figure 4.1.: The common metrics used in clustering and aggregation decision criteria.

mechanisms benefit from each other. As it was mentioned, clustering applies the decision criteria logic found in aggregation schemes, for clusters' organization, while aggregation gains from using the more stable connections among nodes (in-network aggregation), and ultimately uses the Cluster IDs for a reliable and duplicate-insensitive data fusion (hierarchical aggregation). Subsequently, this will make possible the development of more complex protocols, where clustering and aggregation mechanisms are fully concatenated.

In the context of the aforementioned, we consider that aggregation performed within a cluster entity will lead to more precise and less prone to errors data aggregates. On the other hand, clustering based on similarity of dynamic characteristics will lead to more stable connections among nodes organized in cluster structures. In this way, the first step performed by CBAP consists in the cluster establishment, where vehicles moving in mutual proximity/transmission range organize themselves into clusters. The second stage starts while these nodes exchange their information (namely their average velocities), followed by in-network aggregation procedure performed by the Cluster Head (CH). After the aggregate (i.e. cluster average velocity) is calculated, individual clusters can start exchanging their aggregates thus contributing to the hierarchical inter-cluster data aggregation.

By mentioning the concatenation of the two sub-protocols (clustering and aggregation), we mainly refer to the possibility of using a common parameter in the deployment of both algorithms: cluster establishment on the one hand, and decision criterion and information fusion on the other. As it can be seen from Figure 4.1, cluster joining and in-network aggregation decisions use the same metric β, measuring the difference between node and cluster average velocities. On the other hand, both re-clustering and hierarchical aggregation decisions use another metric δ, measuring specifically the difference between aggregates of two adjacent clusters. We also strongly believe that a higher priority should be given to the dynamic characteristics of nodes, rather than their static geographical location parameters, since their dynamics is the main challenge which hinder the establishment of stable and lasting connections.

5

Protocol Model

When designing the CBAP protocol and analyzing its main features and functionality aspects, it is important to understand the complexity of this protocol and distinguish between its two components: clustering algorithm and aggregation scheme. In this respect, the present chapter is divided into four main sections, two of which are dedicated to a detailed description of the CBAP's sub-protocols, and one section deals with the analysis of some functionality case scenarios. The fourth section concludes the chapter.

5.1. Clustering Algorithm

As starting points in the clustering sub-protocol design, we consider that clusters are built among nodes moving in the same direction, fact that ensures a longer contact-time during which nodes are in each other's communication range[1]. Clusters moving in opposite directions should not affect each other, as well as clusters moving in the same direction (sharing the same or adjacent lanes) but with highly different velocities, should not also

[1] The heading of vehicle movement with regard to the true north is specified in one of the CAM message fields. We consider in the following that nodes check these fields for similarity first, before initiating any clustering negotiations. Also, in the generic case (e.g., multiple moving directions, city scenarios), velocities need to be modeled as vectors that take into account the vehicle's heading. In this work however, we consider this specific parameter as a CAM-related feature, therefore, regard velocities as scalars. Nevertheless, the concepts presented in the following remain the same.

mutually interfere. Our clustering algorithm is based on CBLR and is an adaptation of it, introducing a series of new approaches in cluster establishment and maintenance. According to it, nodes periodically exchange beacons, the so-called HELLO messages, containing the vehicle's basic parameters, such as node average velocity, Node/Cluster ID, and so on.

As it was stated in the previous section, we consider the node's dynamic characteristics as the main and the most important parameters necessary being considered during the clustering procedure. In this respect, we propose a new metric for cluster formation, namely the ***relative average velocity (*** v_{rel} ***)***[1] between nodes, which is considered beside the transmission range (i.e. clusters are established exclusively between nodes located in each other's proximity and single-hop transmission range). This metric was chosen in accordance with our working theory, stating that nodes with velocities close in values: $v_i \approx v_j$ (i.e. $v_{rel} \longrightarrow 0$), tend to create more stable and long-termed clusters.

5.1.1. Cluster Formation

The CBAP's clustering sub-protocol has several common features and resembling aspects characteristic to the CBLR's cluster formation mechanism. Specifically talking, we base our clustering algorithm on CBLR in the sense that we start with the same cluster messaging patterns. Nevertheless, our clustering mechanism has a well-defined and different identity from its CBLR counterpart, introducing a series of new approaches in cluster establishment and maintenance. Depending on the states of the adjacent nodes, a particular moving vehicle chooses one of the four possible states:

1. cluster undecided (CU);
2. cluster member (CM), and an extension of it called cluster member gateway (CMGW);
3. cluster head (CH);
4. cluster head pending (CHP).

As it is implicitly clear, the CH node plays the most important role within the cluster structure, being responsible for cluster formation and addition of new members (CMs), cluster maintenance, communication/packet exchange with other CHs, and last but not least the main aspect of CBAP – cluster-based (in-network) and inter-cluster-based (hierarchical) data aggregations.

In order to better understand how clusters are built, the following figure presents the Finite State Machine (FSM) of the clustering algorithm implemented in CBAP. Our FSM

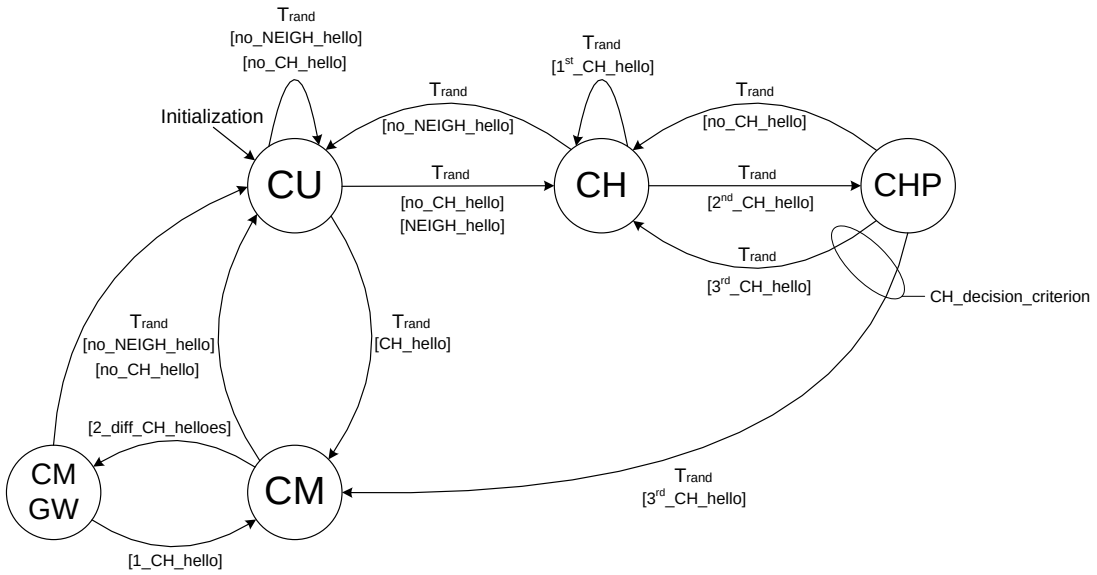


Figure 5.1.: The Finite State Machine (FSM) of the CBAP Clustering Algorithm.

has some common features with the one presented by Souza *et al.* in [21], however it is quite different in its clustering functionality aspects and the way it handles re-clustering.

Cluster formation starts with a particular node entering the Cluster Undecided (CU) state. Basically each node intending to become part of a cluster structure enters this state first. In addition to that, nodes having no neighbors or no cluster membership yet share this state as well. While in the CU state, the node broadcasts a HELLO message (also called beacon) to announce its presence to possible neighbors, in this way requesting the permission to enter a cluster structure. While sending such a beacon, the CU node concurrently starts a timer T_{rand} and waits for any replies, including a HELLO message from a possible adjacent CH. If during this time interval (T_{rand}) the CU node receives a CH_hello packet, it automatically becomes a Cluster Member (CM) of that CH's cluster. In case the CU node gets other types of HELLO messages (i.e. NEIGH_hello from another CU or CM) but the CH_hello, it decides there are no CHs in the sensed neighborhood, so it takes the decision to become a CH. If even no NEIGH_hello packets are received after the expiration of T_{rand}, the UN node sends a new HELLO message, followed by a new timer setting. As long as there are no replies, the CU node preserves its state and sends beacons on a regular basis, with an interval of T_{rand} between transmissions. In this context, the decision to become or not a CH/CM comes only after sensing other nodes in the nearest proximity. This approach in cluster formation differs from the one used by CBLR, where in case the set timer expires before a certain reply is received, the CU node promotes itself to a CH state [16]. This differentiating aspect comes from the consideration, that it is useless to become a CH, while no neighbors are encountered in the node's vicinity.

Another important functionality aspect is the CM's behavior upon the reception of HELLO messages beaconed by adjacent CM or CU nodes. According to our clustering algorithm, if a CM senses other CMs or CUs in its proximity, the beacons received from these nodes are simply discarded. This measure is applied in order to prevent any fluctuations in the cluster stability, thus ensuring the leading role to the CH node.

5.1.2. Velocity Threshold Condition

Introduced in the previous section, the relative average velocity brings a novel concept to today's existing clustering metrics proposed for VANETs so far. In order to make our metric feasible and set the limits within which it can be applied, we introduce the so-called ***Velocity Threshold Condition***. In this case, when two nodes sense each other and start negotiating clustering, they can mutually connect and create a cluster entity only if the *velocity threshold condition* is satisfied. This clustering rule ensures that such nodes have similar or very close velocity values – fact which leads to more durable and more stable inter-vehicle connections. In this respect, when a CU is about to connect to a CH, both nodes first mutually exchange their HELLO packets containing the two Node Average Velocity parameters. Both nodes then perform the comparison of the local and received data, and check if the difference between the two velocity values is less or equal to the preset threshold. As it can be easily seen, this procedure resembles in many aspects the decision criteria found in aggregation schemes, where the aggregate is calculated only if the data components are similar enough.

The connection establishment mechanism is described as follows. Assuming the current velocity is measured every 1 second, an exponential moving average of the past velocity values and the current speed value is constantly computed. Thus, in CBAP the *node average velocity* is calculated based on two velocity values: the *last node average velocity* and the *node current velocity*, instead of only taking the node's current speed into account (the latter being the case of the algorithm used in [22]). Also, a weighting factor α is used to give a higher importance to the latest velocity value. Namely:

$$v_{average_k} = \alpha \times v_{current} + (1 - \alpha) \times v_{average_{k-1}}, \tag{5.1}$$

where: $v_{average_{k-1}}$ is the last calculated node average velocity and $v_{average_k}$ is the newest value of it. In this context, it is important to understand that the most first calculated node average velocity (when $k = 1$) is considered equal to the node's current velocity: $v_{average_k} = v_{current}$, since $v_{average_{k-1}} = 0$ (vehicle has just started its motion).

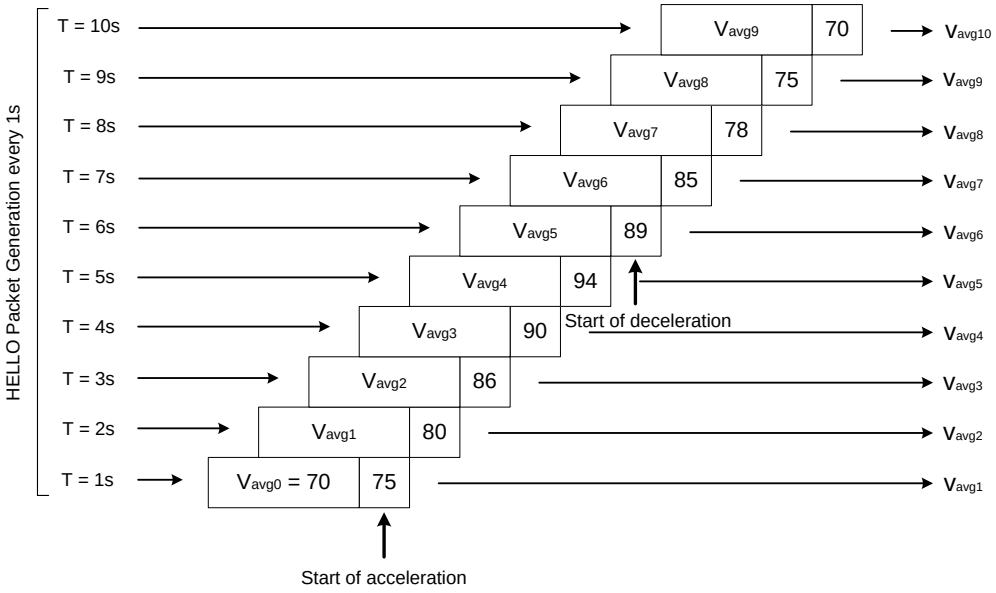


Figure 5.2.: $v_{average(node)}$ calculation algorithm.

As an explanatory example, consider $t \approx 1\,s$ and $\alpha = 0.5$. Now, assume a car is moving with a 70 km/h speed. It decides to accelerate in order to overtake another vehicle moving in front, so each $t = 1\,s$ (every instance a HELLO message is generated) its velocity increases by 5 – 6 km/h. After overtaking is complete, the car starts decelerating reaching its initial 70 km/h speed. Figure 5.2 shows the calculation of *node average velocity* and Table 5.1 presents the corresponding calculation results.

The purpose of this computation algorithm is to ensure that even after short accelerations of a vehicle, its average velocity ($v_{average}$), based on which the node participates in a cluster formation, will not be affected too much. From the presented computation results, it can be seen that despite the node current velocity touches the 94 km/h mark, its maximum average speed reaches a value of only 89.78 km/h, which subsequently decreases during deceleration. If the acceleration however continues and the node current speed keeps growing, the $v_{avg(node)}$ value becomes too high and won't satisfy the *velocity threshold condition* anymore, which consequently determines the CH to refuse the cluster membership request solicited by the new CU node. Therefore, only if the average velocity of a CU node is similar or close to the one of the cluster, the *velocity threshold condition* is met and the node joins the cluster. Namely, a CU joins if:

$$v_{rel} = |v_{avg(node)} - v_{avg(cluster)}| \leq v_{threshold}. \tag{5.2}$$

At this point, unlike proposed in [22], the joining candidate's node average velocity is compared with the average velocity of *all* nodes already participating in the cluster and not

Table 5.1.: Calculation results for $v_{average(node)}$.

v_{avg1}	v_{avg2}	v_{avg3}	v_{avg4}	v_{avg5}
72.50 km/h	76.25 km/h	81.12 km/h	85.56 km/h	89.78 km/h
v_{avg6}	v_{avg7}	v_{avg8}	v_{avg9}	v_{avg10}
89.39 km/h	87.19 km/h	82.59 km/h	78.79 km/h	74.39 km/h

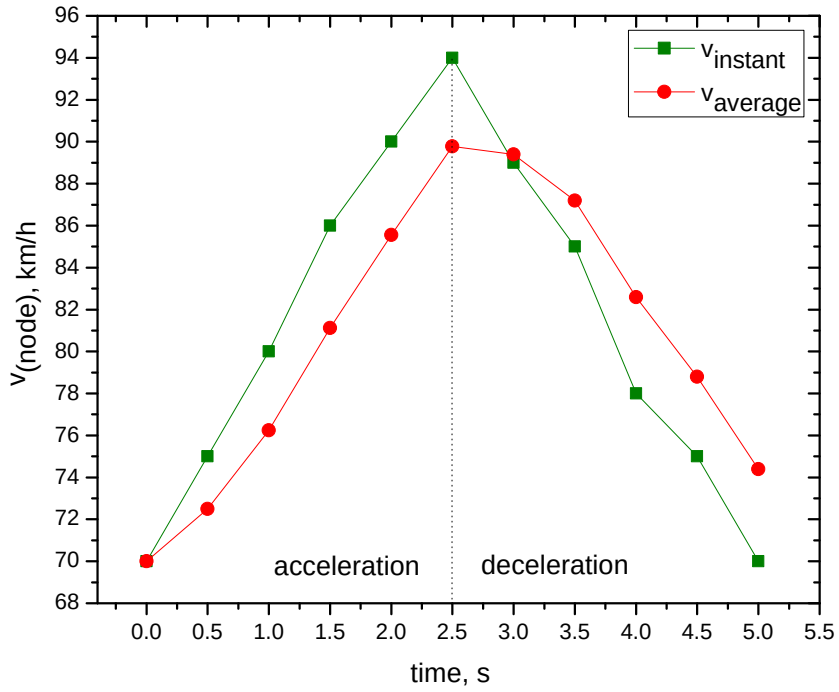


Figure 5.3.: Performance difference between the two clustering algorithms.

solely with the velocity of the CH, therefore, the decision of becoming or not a CM depends to a certain extent on each cluster member:

$$v_{avg(cluster)} = \frac{v_{avg(CH)} + \sum_{i=1}^{m-1} v_{avg(CM)_i}}{m}, \tag{5.3}$$

where m is the total number of cluster members.

As an illustrative comparison, Figure 5.3 shows the performance difference between our average velocity computation algorithm, and the algorithm where only the instant velocities are considered for clustering purposes (such as the one used in [22]).

Last but not least, the threshold parameter $v_{threshold}$ helps the CH to determine the eligibility of the newly received data to be considered or not in the next aggregate computation. The *velocity threshold condition* plays a very important role, because it improves the

clustering algorithm's reliability. This leads to a twofold advantage. First, vehicles moving in the opposite direction will never form a cluster, because their relative velocity is too high and will not satisfy the threshold condition. Second and more important, nodes heading in the same direction, on the same or adjacent lanes but at highly different speeds, will also not join/form a common cluster. This prevents the formation of unstable and most likely easy breakable clusters. The exact value for the velocity threshold condition can be adjusted, thus establishing harsher or milder conditions for joining a cluster. Suitable threshold values will be discussed in Chapter 6.

5.1.3. Packet Structure

For a better understanding of CBAP and its clustering sub-protocol, we present the HELLO message format and the procedure of packet assembling during the cluster formation and negotiation. However, before discussing the packet structure, we have to state that we consider our HELLO messages piggybacked to the CAM beacons. In our consideration, the clustering overhead does not need to be sent at every instance of CAM generation (e.g., for the highway scenario, each 0.1 s), therefore we assume the clustering overhead being piggybacked and disseminated with every tenth CAM message. This results in a $T_{rand} = 1\,s$, just as it was previously introduced. In the following analysis, any reference to packets, beacons or messages is referred to our clustering HELLO message, otherwise, it will be explicitly stated.

Each beacon sent by a node (be it CH, CM or CU) contains four main fields, characterizing that node's cluster membership. These four parameters include: Cluster ID, identifying the cluster that particular node belongs to; Node ID, differentiating vehicles within a cluster; Node State, specifying the node's role in the cluster, and finally Node Average Velocity – parameter calculated according to the earlier described algorithm and updated every time a new HELLO packet is sent. The beacon also contains the Payload field, which is the largest in the packet. The payload's content comprises the useful information shared among nodes, including the aggregate value, or the cluster average velocity ($v_{avg(cluster)}$), as well as such other data types as the number of free parking slots in a parking area, emergency vehicles' messages, road/weather conditions, and so on. For timing purposes and up-to-dateness specifications, our clustering overhead uses two fields: a timestamp and a Time-To-Live (TTL) field. Both parameters will be discussed in more detail in Section 5.3. Figure 5.4 presents the HELLO beacon format. We also specify the approximative length (in Bytes) of each packet's field.

According to CBAP's clustering algorithm, the payload is mainly used for carrying over the data aggregate from the CH to the CMs, thus informing the members about the cluster

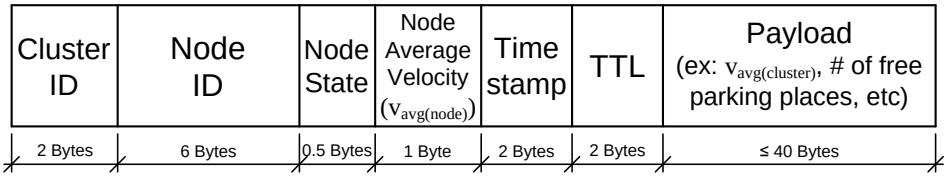


Figure 5.4.: The clustering sub-protocol HELLO packet structure.

average velocity. As it will be shown later, the $v_{avg(cluster)}$ value plays a very important role in both:

1. *CH Decision Criterion*, when two clusters merge and the corresponding CHs have to decide who preserves the CH role and who becomes a CM of the new, merged cluster;
2. Providing information about traffic jams and accidents on the roads.

While considering the packet structure, a special attention should be paid to the beacon message assembling. Initially, when the node starts its activity in the CU state, it is beaconing HELLO messages containing the following values in the packet fields:

1. Cluster ID = 0;
2. Node ID = Node MAC Address;
3. Node State = CU;
4. Node Average Velocity = $v_{avg(CU_node)}$;
5. Payload = $v_{avg(CU_node)}$.

When a node migrates from CU to CH state, its fields' values are automatically changed. The CH sets a random 16-bits long number for the Cluster ID using a Random Number Generator (RNG) and five hash functions (this procedure will be analyzed in more detail in Section 5.3). The Node ID is by default set to the unique MAC address specific to any node. Basically the Node ID is always a constant value and is preserved independently of the state characterizing a moving vehicle. The Node State is set to CH and the Node Average Velocity, as well as the initial Payload are both set to $v_{avg(CH_node)}$. At the beginning of cluster establishment the Payload value can be considered equal to the CH Node Average Velocity, because initially the cluster consists of only one node (namely the CH): $v_{avg(cluster)} = v_{avg(CH_node)}$.

When the first CU tries to connect to the CH, a HELLO packet exchange between the two nodes takes place. Once both nodes hold the own and the opposite node's packets, the comparison: $v_{avg(CH_node)}$ vs. $v_{avg(CU_node)}$ is performed individually by each node. If

the *Velocity Threshold Condition* is met, the CU node sets its Cluster ID to the one learnt from the CH, and consequently changes its Node State from CU to CM. On the other hand, the CH includes the new CM Node ID with the corresponding CM Node Average Velocity entry in its Cluster Table. This procedure is followed by the calculation of the new aggregate, which takes into account the newly connected CM. In this respect, the first exchange of HELLO packets between the CH and CU is mainly intended for connection and cluster membership establishment, but is also concurrently used in the first aggregate calculation. Using the CM Node ID, the CH keeps a list of current average velocities per each node and updates those values if a new report (HELLO packet) from an already known CM is received. Briefly, the CH field values can be expressed as:

1. Cluster ID = # generated by RNG;
2. Node ID = Node MAC Address;
3. Node State = CH;
4. Node Average Velocity = $v_{avg(CH_node)}$;
5. Payload = $v_{avg(cluster)}$.

If the CU node becomes a new CM of the cluster entity, its fields are set to:

1. Cluster ID = # learnt from the CH;
2. Node ID = Node MAC Address;
3. Node State = CM;
4. Node Average Velocity = $v_{avg(CM_node)}$;
5. Payload = $v_{avg(cluster)}$.

In the above mentioned context, when the first CM connects to the CH, the initial aggregate value is calculated as:

$$v_{avg(cluster)} = \frac{v_{avg(CH_node)} + v_{avg(CM_node)}}{2}. \tag{5.4}$$

5.1.4. Cluster Maintenance

After the cluster establishment, nodes (CH and CMs) continue exchanging HELLO messages in order to maintain the cluster entity up-to-date and keep its structure integrity. This is also important for the regular aggregate computation performed by the CH, using the individual information sent by each CM. After the aggregate is calculated, the CH uses its beacons in turn to inform the CMs about cluster average velocity ($v_{avg(cluster)}$).

An important aspect of the CBAP clustering sub-protocol is the cluster maintenance. During this continuous and repeatable stage, the *velocity threshold condition* is constantly checked at each HELLO packet exchange between the CH and its CMs. If this condition is not met for a certain exchanged beacon between CH and a CM, the latter will have to consider leaving the cluster. In such a case, in order to preserve the cluster's integrity and structural stability, even when $v_{rel} > v_{thresh}$, a $3 \times T_{rand}$ interval is given to the member node to satisfy the threshold condition again. During this time lapse the CH does not consider the too high/too low $v_{avg(CM_node)}$ in the aggregate ($v_{aver(cluster)}$) calculation. If the threshold condition is still not fulfilled at the expiration of $3 \times T_{rand}$, the CM leaves the cluster thus changing its state to CU, while the CH deletes this member from its Cluster Table.

One important feature that our clustering sub-protocol has lent from CBLR is the sustenance of two table entities, necessary in cluster maintenance: Cluster Table and Cluster Neighbor Table. The Cluster Table, mentioned in the previous sections, contains a list of Node IDs belonging to the CMs participating in cluster formation. Each Node ID is associated with a special entry corresponding to the Node Average Velocity ($v_{avg(node)}$). At each exchange of HELLO messages, the CH updates this table, consequently using the newly received data in aggregate computation. The Neighbor Cluster Table, on the other hand, contains a list of Cluster IDs and their corresponding aggregate ($v_{avg(cluster)}$) values characterizing the adjacent clusters. This information becomes available to the CH by means of Gateway CMs (CMGWs), who in turn participate in aggregates exchange between CHs of neighboring clusters. This procedure will be discussed in more detail in Subsection 5.1.6 of this chapter.

5.1.5. Re-clustering Mechanism

An important aspect of the clustering algorithm is the way it handles re-clustering, i.e. when two CHs enter each other's communication range (thus mutually sensing one another) and start considering merging their clusters. According to our re-clustering mechanism, when a CH receives a first CH_hello message from another CH, it takes no actions – that beacon is considered a random one and is simply discarded. Such a situation can happen when a CH encounters another CH heading in the opposite direction. This behavior expressed by disregard of the first HELLO packet is intended to prevent over-eager re-clustering upon an accidental contact of two CHs. In case the second CH_hello message is received, the CH enters a special state, called Cluster Head Pending (CHP). While in this state, the CH takes no actions, instead continues listening to the medium. The CHP state is basically conceived as a second measure taken to prevent accidental re-clustering, hence, instability

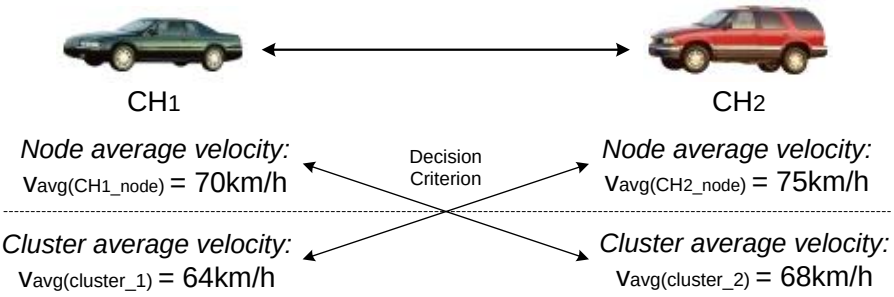


Figure 5.5.: CH Decision Criterion.

of the cluster structure. If the third CH_hello is not received within the T_{rand} period, the CHP returns back into CH state and keeps functioning as before (see Figure 5.1). On the other hand, if the third CH_hello beacon is received, the so-called ***CH Decision Criterion*** comes into action (Figure 5.5). However, before this criterion can be applied, both CHs make first a short comparison of their cluster average velocities and check whether these two parameters satisfy the *velocity threshold condition*:

$$|v_{avg(cluster_1)} - v_{avg(cluster_2)}| \leq v_{thresh}. \tag{5.5}$$

Only if these two values are close enough to each other, and the threshold condition is fulfilled, the *CH Decision Criterion* is applied. This comparison is performed in order to ensure that the two clusters' characteristics are similar enough for both entities to be merged, as well as to guarantee that both CHs, and respectively their clusters, move slowly enough relative to each other. Thus, in case of merging the two structures, the newly formed cluster should be as stable as possible. As it will be shown later, the same approach used in re-clustering, is further applied in the hierarchical aggregation scheme – another particularity which makes clustering and aggregation strongly concatenated.

In order to decide which node will continue playing the CH role and which node switches its state to CM of the newly merged cluster, both CHs compare their node average velocity with the opposite cluster average velocity:

$$\delta_1 = |v_{avg(CH1_node)} - v_{avg(cluster_2)}|, \tag{5.6}$$

$$\delta_2 = |v_{avg(CH2_node)} - v_{avg(cluster_1)}|. \tag{5.7}$$

In this situation, cluster average velocity is a very helpful parameter, because it characterizes the dynamic/mobility properties of the structure as a whole. The contest is won by the CH whose node average velocity is closer to the opposite cluster average velocity. Thus, in case $\delta_1 < \delta_2$, CH1 preserves its state and becomes the CH of the new, merged

cluster, while CH2 changes its state to CM. Otherwise, CH2 becomes the new CH. With a quite small probability it can happen that $\delta_1 = \delta_2$. If this is the case, both CHs compare their randomly generated Cluster IDs. The CH with a lower Cluster ID number wins the contest and preserves its CH status. This mechanism will be dealt with in more detail in Section 5.2. The *CH Decision Criterion* can be algorithmically expressed as:

1: **if** $|v_{avg(CH1)} - v_{avg(cluster_2)}| = min$ **then**
2: $CH2 \Longrightarrow CM1$
3: **else**
4: **if** $|v_{avg(CH2)} - v_{avg(cluster_1)}| = min$ **then**
5: $CH1 \Longrightarrow CM2$
6: **end if**
7: **end if**

After the new CH was reelected, the CH who lost the contest becomes a new CM and changes its Cluster ID to the one learnt from the winner CH. Subsequently, the new CM sends an update beacon to all its previous CMs and announce them to change their Cluster IDs as well. Thus a new, larger cluster is formed. After such a merging of two clusters, most probably not all the CMs of the losing CH will be in the transmission range of the newly elected CH. This means that those CMs will simply lose their cluster membership, thus changing their state to CU. In case there are more than one such nodes, they will automatically participate in the formation of a new neighboring cluster.

5.1.6. Inter-cluster Communication

Inter-cluster communication is only possible if there is at least one CM Gateway (CMGW) node in one of the two neighboring clusters. This is a special type of CM simultaneously sensing two or more CHs. The CHs might be out of each other's reach, but a CMGW can sense both of them (the hidden station phenomenon). Basically, once a CM node starts sensing two CHs in its vicinity, it automatically changes its state to CMGW. Different from CBLR, in the CBAP clustering sub-protocol, the CMGW has to decide to which cluster it will stay further connected. We called this algorithm ***CH Selection Decision***, where the velocity values of the CMGW versus both sensed CH1 and CH2 have to be considered (Figure 5.6). The two pairs are compared, and the lowest relative velocity decides to which cluster the CMGW stays attached.

When the CMGW sends a regular HELLO message to its CH, that beacon also includes the Cluster ID and the aggregate ($v_{avg(cluster)}$) of the neighboring cluster, in its payload field. It is important to emphasize that according to CBAP, CHs can not address packets and send specific *unicast* beacons to the adjacent CHs. CBAP implements only a broadcast

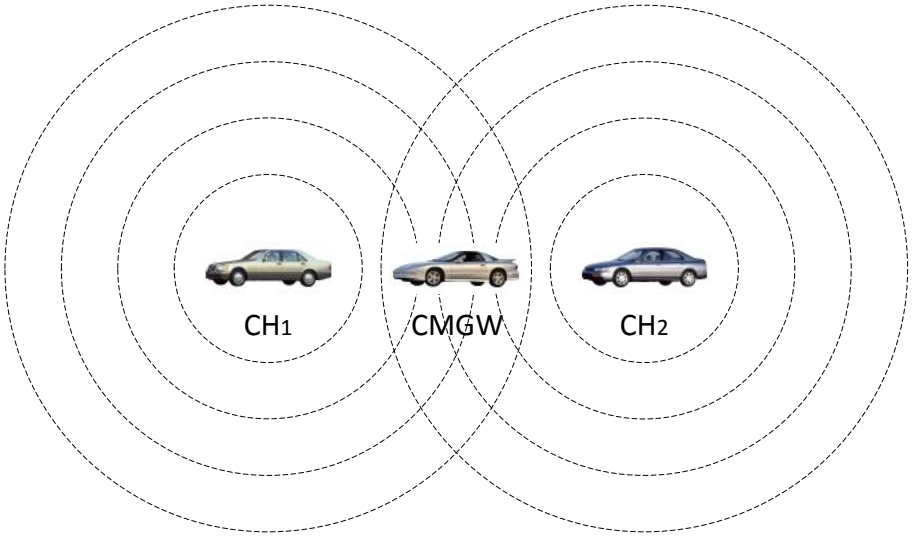


Figure 5.6.: CH Selection Decision.

dissemination mechanism and no specific routing algorithms are foreseen. In addition to that, the CMGW undertakes the task of exchanging HELLO messages (containing valuable information) between neighbor CHs. With that said, a CH does not differentiate between a CMGW and a regular CM. On the other hand, if a specific routing protocol is configured atop CBAP, the CH will have to mark the CMGW in its cluster table and ultimately use that connection for unicast inter-vehicle data transmissions.

The algorithm of the *CH Selection Decision* can be expressed as follows:

1: **for** $v_{rel1} = |v_{avg(CH1)} - v_{avg(CMGW)}|$ &
2: $v_{rel2} = |v_{avg(CH2)} - v_{avg(CMGW)}|$ **do**
3: **if** $v_{rel1} < v_{rel2}$ **then**
4: $CMGW \longrightarrow CMGW1$
5: **else**
6: **if** $v_{rel1} > v_{rel2}$ **then**
7: $CMGW \longrightarrow CMGW2$
8: **end if**
9: **end if**
10: **end for**

In this context it is worth emphasizing that hierarchical aggregation performed between adjacent clusters is only possible if there is at least one CMGW node in one of the two neighboring clusters. These nodes play the role of inter-cluster bridges, necessary to effectuate aggregates exchange between clusters, thus making the subsequent hierarchical aggregation possible. This exchange of dynamic data can be very helpful and informative for nodes moving behind, because this information provides an insight into the future

expected traffic situations. However, in the capacity of payload exchanged among clusters, different data types, supporting applications other than improved driving can also be used.

5.2. Clustering Case Scenarios

There are a few interesting case scenarios related to clustering sub-protocol functionality that are worth being considered. The following three bullet points present a short analysis which provides a further understanding of some important features characterizing the cluster formation procedure.

1. Let us assume that a new CU node joins the cluster for a short period of time and then leaves it. Since every HELLO message is transmitted periodically, its main purpose is to provide nodes with up-to-date information related to cluster state. After the new CU connects to the CH, the CH calculates the new aggregate, thus taking into account the freshly connected CM. Despite the new CM joins the cluster for a short time span and then leaves it, the aggregate will not be affected or get inaccurate, since the CU becomes part of the cluster only if it satisfies the velocity threshold condition. Hence, if the new node meets that condition, its average velocity is close (or even equal) to the cluster average velocity, therefore the new aggregate will not differ too much from the old one. This proves to be another advantage of the *velocity threshold condition*. When the newly connected node leaves the cluster, the CH waits for a $3 \times T_{rand}$ interval and then deletes the disjointed node from its Cluster Table, followed by the recalculation of the new aggregate value. The same procedure is taking place when a permanent CM leaves the cluster.

2. Another important clustering scenario is the case of a CH leaving the cluster. This situation can occur when the CH takes another path or pulls over, while the rest of the cluster keeps moving in the initial direction. If so, all CMs preserve their cluster membership for a $3 \times T_{rand}$ time span and then, if still no CH_hello beacons are sensed, change their states to CU. In this case, nodes start negotiating the formation of a new cluster and a new CH reelection. This situation is analyzed in the following third case scenario.

3. If a few CU nodes (driving with similar or close mobility patterns) come into mutual contact, most probably it will happen that a few of them, after sensing each other, will simultaneously (within the same T_{rand} time slot) change their states to CH. If so, normally after three exchanged beacons, the multiple CHs will decide to merge their single-node clusters, by applying the CH Decision Criterion. However in this case, the CH Decision Criterion can prove to be useless, because it works properly for clusters consisting of at least two nodes. If the cluster contains only one node (namely the CH), the simple comparison $v_{avg(CH)_A}$ versus the opposite $v_{avg(CH)_B}$ will lead to the same difference results,

i.e., $\delta_1 = \delta_2$. In this case, as it was previously mentioned, the CHs compare their Cluster IDs. The CH with the lowest generated Cluster ID wins the contest, and all the adjacent CHs with higher Cluster IDs change their node states to CM.

5.3. Cluster-based Aggregation Scheme

The CBAP protocol implements a new approach in both in-network and hierarchical data aggregation schemes, where clustering and aggregation mechanisms are strongly linked and concatenated. As it was shown, characteristic to CBAP is that the aggregate value plays a very important role in the cluster formation and maintenance procedures, namely when a node joins and stays attached to a cluster, while the velocity threshold condition is continuously checked and the cluster average velocity is repeatedly computed. Further the aggregate is extensively used in the re-clustering mechanism and its *CH Decision Criterion*, as well as in the hierarchical aggregation decision. Similar to other protocols, CBAP uses the average velocity as aggregate parameter, however its cluster-based nature introduces a novel concept to today's aggregation schemes.

In the following, the procedures of in-network and hierarchical aggregation schemes are further analyzed and discussed.

5.3.1. In-network Aggregation

In-network (or intra-cluster) aggregation scheme consists in the data aggregation procedure deployed within the cluster by simply calculating the average velocity of the entire group of cluster member nodes. This value is repeatedly computed by the CH (5.3), based on the individual node average velocities ($v_{aver(node)}$) disseminated and reported by each CM. The continuous calculation of $v_{aver(cluster)}$ ensures that each time a new node joins the cluster, its data will be considered in the new aggregate computation, as well as guarantees that the aggregate value is constantly kept up-to-date. The main aim of in-network aggregation is to ensure that the cluster entity disseminates a single aggregated datum characterizing its structure, instead of tens or even hundreds of individual data generated by participating nodes. In this respect, the saving of bandwidth resources becomes the primordial motivation behind.

The cluster average velocity is a very informative parameter, playing a valuable role in traffic congestion management, since the average velocity can tell a lot about the group of vehicles it characterizes. In this respect, if $v_{aver(cluster)}$ for a highway region is dramatically

low, this indicates a traffic jam or an accident. On the other hand, an average/high value shows that the road is idle and vehicles can move without encountering certain obstacles.

As it was stated in this section's introduction, the intra-cluster aggregate plays a crucial role in the operation of our clustering mechanism. Because it was extensively analyzed in the clustering-related sections of this work, in the following we will not further discuss in-network aggregation, instead focus on another important functionality aspect of CBAP, namely, the hierarchical aggregation scheme.

5.3.2. Hierarchical Aggregation

The hierarchical (or inter-cluster) aggregation scheme is applied among adjacent and distant clusters, where similar and close in values aggregates are disseminated and further hierarchically aggregated. This method has the same objective and purpose as an in-network aggregation scheme – namely, a rational use and saving of bandwidth radio-resources, however on a larger scale. Moreover, an important motivational aspect behind the use of hierarchical aggregation is the limited knowledge provided by in-network aggregates, due to the bounded 1-hop dissemination range within which our clustering mechanism works. This means that intra-cluster aggregates offer only little knowledge about the upcoming traffic situations, therefore, an extension of this vision is extremely crucial. The knowledge regarding the expected traffic scenarios, that originates from neighboring and distant clusters is indispensable in this regard. Hence, hierarchical aggregation introduces a mechanism for combining the data tuples of local and foreign origins, in order to build an accurate model of the external world.

In the following we discuss the basic conceptual ideas of our hierarchical aggregation scheme, and subsequently focus our attention on a detailed analysis of the main features and challenges characterizing the inter-cluster aggregation. A novel timing paradigm is also proposed in this section. The main functionality aspects related to duplicates identification and their annihilation, as well as aggregates fusion mechanisms will be further covered in Chapter 6.

To provide a better insight into the hierarchical aggregation scheme, the following scenario is considered. Let us assume that two neighboring clusters are driving in mutual proximity, so that a CMGW can sense both CH nodes (Figure 5.7). Once the CMGW node receives both aggregates of the adjacent clusters, it makes an exchange of the two tuples (as discussed in Subsection 5.1.6), as a result of which CH1 receives the aggregate $v_{aver(cluster2)}$ and CH2 gets the aggregate $v_{aver(cluster1)}$. Once each CH holds both aggregates, a comparison of the two values is performed. At this point, if the two

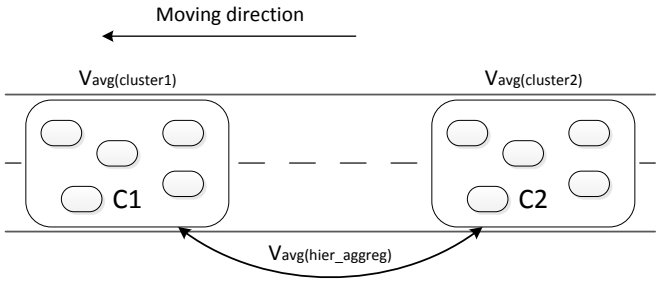


Figure 5.7.: Hierarchical aggregation model.

tuples are comparable and their difference does not exceed a preset threshold (we use the same value applied in the *velocity threshold condition*, though it can be set to a different value):

$$|v_{avg(cluster1)} - v_{avg(cluster2)}| \leq v_{thresh}, \tag{5.8}$$

the hierarchical aggregation is executed:

$$v_{avg(hier_aggreg)} = \frac{v_{avg(cluster1)} + v_{avg(cluster2)}}{2}. \tag{5.9}$$

The comparison of the two aggregates helps to determine their similarity degree, thus making possible to estimate the amount of accuracy lost, if the two tuples are hierarchically aggregated. In this respect, the threshold stands for the maximum tolerable value, for which hierarchical aggregation is a reasonable solution and accuracy is within admissible limits. Of course, as long as the threshold condition is satisfied in one cluster, it will be met in the adjacent cluster as well. As a result, this makes both CHs compute the same hierarchical aggregate, thus disseminating a single data value.

It can however happen that the difference between the two aggregates exceeds the threshold value. This means that both data structures contain a too different information, which needs being kept separate. In these circumstances, while disseminating beacons within the cluster, both CHs broadcast their own local aggregate, as well as the adjacent cluster's aggregate stored in the Neighbor Cluster Table. This ensures that CMs get accurately acquainted with the traffic situation in the neighboring clusters, which in the ideal case might span long distances ahead. This process of aggregates fusion or their separate transmission continues hierarchically with more distant clusters, until the information becomes irrelevant (dies out) or is discarded due to extra-bandwidth use.

It will be shown later that two particular parameters, namely the *Time-To-Live (TTL)* and the *Maximum Admissible Transmission Margin (M)* are specifically designed to cope with

the two mentioned aggregation flaws, i.e., information irrelevance over time and running out of bandwidth resources.

5.3.3. Challenges and Solutions

One of the major challenges characteristic to hierarchical aggregation is the duplicate-sensitivity problem, when a certain tuple or even several tuples might repeat multiple times in the final hierarchical aggregate. Of course, this flaw prevents the obtaining of highly accurate aggregates and makes hierarchical aggregation prone to errors and loss of accuracy. One quite famous method for coping with this drawback was proposed by Lochert *et al.* [26], consisting in a probabilistic technique used to aggregate the disseminated data. This mechanism does not perform aggregation of the actual values, but uses instead a modified Flajolet-Martin sketch [27] as a probabilistic approximation for the values. Their protocol is basically used in the estimation of the number of free parking places, and its main advantage is the fact that aggregates are duplicate insensitive. This means, when multiple data items are available for the same region, they can be combined in such a way that the final aggregate will contain all the initial informations. In the parking conditions this results in aggregates that can be combined or re-combined without counting the number of free slots multiple times. According to [26], upon reception, the local and received sketches representing the data aggregates are merged using the bitwise OR operation. As it can be easily seen, bits that have once been set to one will never get unset again, thus the sketch is only able to follow increasing values. In order to overcome this limitation, the authors adjust their algorithm implementing the so called soft-state sketches, where instead of using single bits at each sketch's index position, small counters of *n* bits length are deployed. These counters are represented by a time-to-live (TTL) value in the range of $0, ..., 2^n - 1$. As a result, the operation of setting a bit to one after an observation is replaced by setting the corresponding counter to the maximum TTL, i.e., $TTL = 2^n - 1$. Before a certain sketch is disseminated using broadcast packets, all counters in the locally maintained sketches are decremented by one (if they are not equal to zero yet) [26].

Despite of the several advantages brought by Flajolet-Martin sketches and their quite robust implementation proposed by Lochert *et al.* in coping with duplicate-sensitivity problem, there are two significant limitations that prevent the use of this algorithm in our hierarchical aggregation scheme. The main issue which arises in this context are the dynamic characteristics of our aggregates, namely the cluster average velocities. With this respect, node and cluster average velocities are much more dynamic and varying in nature, than the number of free parking places aggregated in [26], which in turn are characterized by more static features. Secondly, an additional arising problem is that in [26] aggregates

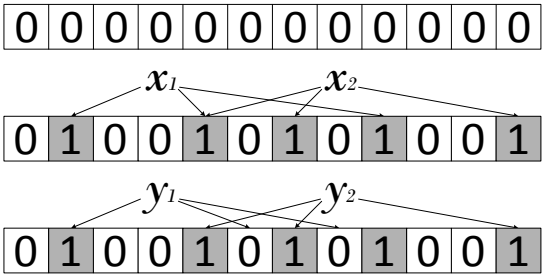


Figure 5.8.: An example of a Bloom filter and its operation [28].

are counted and added one by one, however in our scheme we use averaging (summation and division of two or more sketches) – operation that increases the potential errors. In this context, we developed another technique for data duplicates checking and identification, based on a special data structure, called Bloom filter.

A Bloom filter is a probabilistic, simple-space randomized data structure for representing a set intended to support membership queries [28]. A test based on Bloom filtering can issue two possible results, either "Element possibly in the set", or "Element definitely not in the set". The first result can be associated with two particularities, one corresponding to the real presence of the searched element in the set, and the other related to a *false positive* match, where the searched element is identified as being part of the set, even though it is not. The second result, if issued, is always correct, since false negatives are not possible in Bloom filters [29], [30].

A Bloom filter representing a set $S = x_1, x_2, ..., x_n$ of n elements is conceptually a bit array of *m* bits, initially all set to 0. A Bloom filter also uses *k* different, independent hash functions $h_1, ..., h_k$ with range $1, ..., m$ defined, each of which being responsible for mapping/hashing a certain set element x to one of the *m*-array positions, with a uniform random distribution. To add an element to a set, first it has to be fed to each of the *k*-hash functions to get *k* array positions, where the corresponding bits are set to 1. Then the obtained bit array is added to the set using the bitwise-OR operation, thus an array coordinate can be set to 1 multiple times, but only the first change has an effect [28]. In order to check if an item y is in S, it has first to be hashed, then it is checked whether all $h_i(y)$ are set to 1. If not, then definitely y is not a member of S. If yes, y is supposedly concluded being in the set. Figure 5.8 explains the Bloom filter's functionality by setting an example, where two elements x_1 and x_2 are added to the common set S (Bloom filter) using $k = 3$ hash functions. When other two elements y_1 and y_2 are checked whether they are in the set, both items are first hashed using the same hash functions, and then their array coordinates are analyzed. As it can be seen, $y_1 \notin S$, while $y_2 \in S$, thus, if no other elements were added before, it can be concluded that $y_2 = x_2$.

5.3.4. Duplicate Detection Algorithm (DDA)

In the context of data-duplication problem, the cluster-based hierarchical aggregation scheme introduced in our CBAP protocol includes two major stages:

1. When two or more aggregates are exchanged between two adjacent clusters, before hierarchically aggregating them, both CHs check the threshold condition (5.8).

2. If the difference between the two aggregates satisfies the preset threshold, the *Duplicate Detection Algorithm (DDA)* is further applied, in order to determine whether the two data structures contain certain common tuples, that could have led to duplicates presence.

The *Duplicate Detection Algorithm* assumes some important aspects. First of all, each initial in-network aggregate calculated by individual CHs is associated with the originating Cluster ID. For efficiency purposes, the Cluster ID linked with the aggregate can be regarded as a valid Bloom filter, that is basically a 16 bit-long binary array ($m = 16$), which is obtained using $k = 5$ hash functions and is generated by means of a Random Number Generator (RNG). This results in 16 bit-long arrays containing preponderantly logical 0s and only 5 logical 1s, with a random position within the array. In this context, it is important to emphasize that the Cluster ID plays the role of a fingerprint, necessary for aggregates' identification.

When a CH holds two aggregates, it has to consider whether to hierarchically aggregate them or disseminate both tuples apart. If the earlier mentioned threshold condition is fulfilled, both Cluster IDs associated with the two aggregates are further analyzed, checking them against duplicates presence. The two Cluster IDs could already contain other potential Cluster IDs previously fused, therefore, it makes sense to determine the intersection of the two possible sets of Cluster IDs that contributed to the two items that are about to be aggregated. In this context, the main goal is to determine the number of items in the set intersection of the two Cluster IDs, which is equivalent to the number of duplicates. Assume *X* and *Y* are the two Cluster IDs, whose intersection value (i.e. the number of common elements) is necessary to determine. The Bloom filter properties specific to our Cluster IDs, offer a way to estimate the number of elements contained within each item (Cluster ID) apart, namely:

$$X^* = -\frac{N \times \ln\left(1 - \frac{X}{N}\right)}{k}, \tag{5.10}$$

where: N is the length of the filter $(N = 16)$, k is the number of hash functions per Cluster ID $(h = 5)$, and X is the number of bits set to 1 [31].

Using further the Bloom filter logic and basic operations, we extend (5.10) to a set join:

$$X^* \cup Y^* = -\frac{N \times \ln\left(1 - \frac{X \cup Y}{N}\right)}{k}, \tag{5.11}$$

where: $X \cup Y$ is the number of bits set to 1 in either of the two Cluster IDs (or in other words, the total number of bits set to 1 in the Combined Cluster ID (CCID)[2]) [31]. *Note: the CCID can be obtained by combining the Cluster IDs using the bitwise-OR operation.*

Consequently, by extending (5.11), the intersection of the two items (possible sets) *X* and *Y* is calculated as:

$$X^* \cap Y^* = X^* + Y^* - X^* \cup Y^*, \tag{5.12}$$

where: X^* and Y^* are the estimates (approximations) of the number of items contained within both Cluster IDs (Bloom filters) (5.10).

Because the value of X^* is an approximation of the number of items (Cluster IDs) contained within the CCID, it has to be always rounded before performing further calculations. The rounding operation is applied for compensation of possible errors, introduced due to the approximating nature of Bloom filter operations. Thus, for values exceeding $X^* \geq 0.5$, X^* is rounded up to 1. If $X^* < 0.5$, X^* is rounded down to 0. Now, if the value of the calculated intersection: $X^* \cap Y^* \geq 1$, this means that both Cluster IDs (CIDs) contain one or multiple similar CIDs, hence, the hierarchical aggregates they represent also include common duplicates. In this context, if the duplicate detection algorithm returns a result of minimum 1 repeating tuple, an error is issued and the two aggregates are not further hierarchically aggregated, instead both are separately disseminated, preserving their initial CIDs. On the other hand, if the intersection results in a value: $X^* \cap Y^* < 1$, this means that both CIDs do not contain any duplicates, therefore the hierarchical aggregation of their tuples can be executed. If so, the average of the two aggregates is computed (5.9), and the newly calculated CCID is associated with the hierarchical aggregate.

An interesting aspect worth mentioning is that we set the threshold value for the DDA algorithm equal to 1, i.e., no duplicates are tolerated and even the detection of one single duplicate results in prohibition of hierarchical aggregation. This is a trade-off between data accuracy/quality and bandwidth resources use. In this work, we opted for the highest data quality, however this also results in a higher consumption of bandwidth resources – drawback that we try to solve using data truncation (analyzed in the following). In this

[2] In order to differentiate between unitary Cluster IDs (specifying single clusters) and complex Cluster IDs containing multiple, previously combined Cluster IDs, in the following we refer to complex Cluster IDs with the generic term "Combined Cluster IDs" (CCIDs). In some contexts however, both terms are used interchangeably, because a CH analyzing two CIDs/CCIDs does not differentiate between the two, because their length is always the same ($m = 16$).

respect, higher DDA thresholds are conceivable, i.e., if the threshold is raised (say up to 3, or 4), small amounts of duplicates will be tolerated, fact which results in a lower data quality and precision, but higher bandwidth savings (because more aggregates are hierarchically aggregated).

A possible scenario that can happen is when one of the two compared Cluster IDs is fully contained in the other one. In this case, instead of disseminating both data structures, it makes sense to disseminate solely the largest one. Therefore, both CIDs undergo further a bitwise-AND operation. This second checking-stage of the DDA algorithm is intended to improve the result obtained by means of the bitwise-OR. Thus, if the bitwise-OR issues a result ≥ 1 and the bitwise-AND returns one the two involved CIDs, this means that the returned CID is a duplicate which is fully comprised within the other CID. If this is the case, the duplicate is discarded and the larger CID is further disseminated.

5.3.5. Timing and TTL

In the context of data aggregation, timing is a very crucial aspect associated with the fused data. Traditionally, many aggregation schemes [23], [24] use aggregate-attached timestamps, that provide the exact generation time of the data tuple. When a node receives two similar aggregates, it checks their timestamps and discards the oldest aggregate. One drawback specific to this timing method is that an outdated aggregate can exist and be disseminated within the dynamic network for a quite long time, even when the traffic situation it describes has already passed. Moreover, because of its relativistic nature, an aggregate associated with a timestamp needs to have newer counterparts (with a newer timestamp) to whom it can be compared. In order to solve these few drawbacks, we introduce the *Time-To-Live (TTL)* parameter, which defines the amount of time a HELLO packet is actual/relevant and valid. Basically, the TTL is used besides the beacon's generation timestamp. Each time a node receives a new packet, it checks whether: $T_{local} - T_{generation} \leq TTL$. If this condition is satisfied, it means the aggregate is valid and can be used for further operations. If however the difference between the local and the beacon's generation times exceeds the TTL, the packet is declared outdated and consequently discarded. Timing is essential in this context, because it ensures that the outdated aggregates die out and are no longer in use. The TTL variable in our interpretation is a function of two parameters:

$$TTL = \frac{R_c}{v_{avg(cluster)}}, \tag{5.13}$$

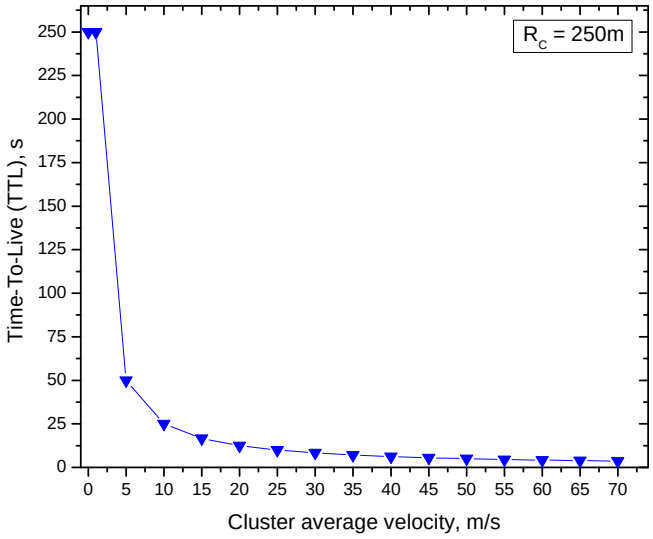


Figure 5.9.: The TTL range as a function of $v_{avg(cluster)}$.

where: $R_c = 250m$ is the sensed communication radius and $v_{avg(cluster)}$ is the cluster average velocity at the time of beacon generation. We consider the sensed transmission range because this is the distance from which the knowledge about surrounding and expected situations originates. We also find it important to take into account the aggregate value in the TTL computation, since TTL is characterizing the currentness of that data structure. With that said, the TTL parameter has a varying nature, and its length is inversely proportional to the value of $v_{avg(cluster)}$. Slowly moving clusters will generate aggregates with longer lifetimes (TTLs), fact which leads to a longer lasting validity of the transmitted data. This feature is particularly important to guarantee that information related to a traffic jam or a car accident will be disseminated as far as possible.

The advantage brought by setting a TTL is twofold:

1. First it ensures that the aggregate will be used only for a short period of time (i.e., as long as it is relevant), and then discarded, when it is no longer informative. In order to update the information contained by the aggregate, new packets with new TTLs will be generated and disseminated. As a result, this feature guarantees that the aggregates are continuously updated and only the newest information is disseminated.

2. Second and quite important, the TTL will be a reasonable solution for coping with the risk of getting a full 1-bit CCID array, upon the combination of several Cluster IDs in the hierarchical aggregation procedure. Despite this risk is also considered when generating the Cluster IDs (by means of a preestablished number of hash functions), the TTL could further mitigate this risk.

When two aggregates are further hierarchically aggregated, their Cluster IDs are combined using the logical OR operation and the TTL is calculated based on the new hierarchical aggregate value. On the other hand, if two similar aggregates (with similar CIDs/CCIDs) are received by a node, the aggregate with the newest generation timestamp will be considered for further fusion/dissemination, while the other one is simply discarded.

Last but not least, a car that has been involved in an accident or a heavy traffic jam and stands still, will be characterized by a 0 km/h velocity. If this is the case, the TTL should be set to a default value, since it is impossible to compute it mathematically. In this case, $TTL = 250\,s$, which is similar to the TTL set for a 1 m/s velocity.

In order to present the range of values a TTL can get, Figure 5.9 shows the dependence $TTL = f(v_{avg(cluster)})$.

5.4. Conclusions

The present chapter was dedicated to the proposal and a detailed presentation of our CBAP protocol. As it was introduced earlier, the CBAP is mainly consisting of two sub-protocols – one dealing with organization of vehicles into clusters, and the second organizing in-network and hierarchical aggregations atop the prebuilt cluster entities. In this context, we have primarily focused on discussion, description and analysis of the main features and functionality aspects characterizing the two sub-protocols.

We have introduced a novel approach for nodes' clustering in vehicular networks, based on the similarity degree of vehicles' dynamic characteristics. In this respect, a new clustering metric was proposed, namely, the *relative average velocity* between nodes. We believe that nodes displaying similar or close average velocities, tend to create more stable and longer lasting clusters. In particular, we draw from decision criteria applied in in-network aggregation schemes, in the sense that just as in aggregation mechanisms, where data items have to be similar enough in order to be aggregated, we only cluster nodes if their dynamic patterns are similar enough. Compared to existing approaches for velocity-based clustering, our protocol improves cluster stability by considering the node average velocities, instead of taking their instant speeds into account. It has been shown that such an approach proves to be extremely beneficial for annihilation of instability effects caused by short-termed accelerations related to different dynamic maneuvers.

When discussing our clustering algorithm, it was shown that in-network aggregation has a strong impact on the cluster formation and maintenance procedures. This was particularly possible due to our aggregate consideration ($v_{avg(cluster)}$) in clustering organization, namely applied in cluster joining and re-clustering decisions. This made it possible to achieve one

of our main goals – that of integration and concatenation of both algorithms into one single seamless mechanism.

Last but not least, a special attention has been paid to the proposal of a novel hierarchical aggregation scheme, based on the application of Bloom filters. The new concept is related to deployment of Cluster ID parameters in in-network/hierarchical aggregates' identification. These IDs linked to the aggregated tuples are especially valuable in checking data against duplicates presence, and a subsequent duplicates resolution. As a result, we introduced the so-called *Duplicate Detection Algorithm*, which is pretty robust and efficient in terms of duplicates identification (as it will be shown in the upcoming analysis). We also mentioned about the main limitations that might arise in this context, mainly the probability of false positives and the obtaining of full 1-bit arrays. Nevertheless, it will be further discussed how both problems can be easily solved by applying longer Cluster IDs (a 32-bit alternative), adjustment of the number of hash functions, and the introduction of a TTL field in the clustering overhead.

In the following evaluation chapter, we present a few case studies in order to provide a thorough analysis and a better insight into our hierarchical aggregation scheme's functionality. We also show the implementation of our Bloom filter-based algorithm intended to cope with duplicate-sensitivity problem, as well as introduce a series of new approaches meant to enhance the hierarchical aggregation mechanism for a more reasonable bandwidth consumption and an accurate representation of information relevance.

6

Protocol Evaluation

This chapter is dedicated to a detailed analysis of the two CBAP sub-protocols introduced earlier. The first section covers the evaluation of our clustering algorithm and its expected performance, as well as presents a further comparison with other clustering protocols. As far as the hierarchical aggregation scheme is concerned, the second section gives a thorough evaluation of its functionality, introducing the novel approaches used in duplicates detection and data truncation mechanisms.

6.1. Clustering Algorithm Analysis

In this section, the evaluation of CBAP clustering sub-protocol is conducted from the perspective of three main performance aspects characterizing the cluster formation mechanism. The first aspect refers to the *average expected contact time* between vehicles moving in the same direction and grouping into clusters according to their node average velocity. The goal is to subsequently choose a reasonable velocity threshold value for our clustering sub-protocol. It is obvious that the inter-vehicle contact time has a direct impact on the *cluster stability* – another important parameter characterizing the performance of a clustering algorithm. As a second featuring aspect, the *expected cluster stability* is further analyzed, by deploying a comparison of the CBAP velocity-based clustering versus the

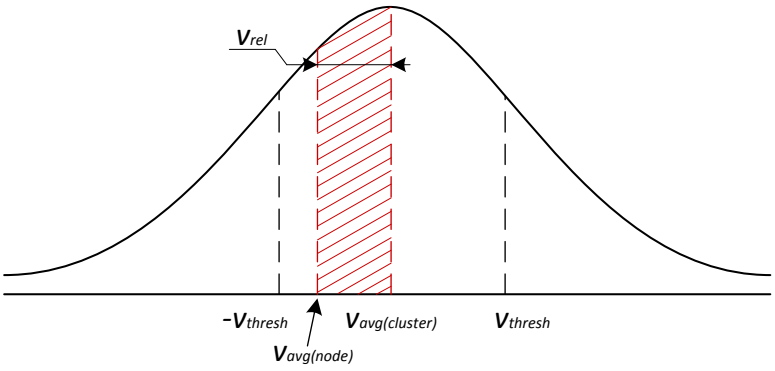


Figure 6.1.: Node satisfying the velocity threshold condition and participating in cluster formation.

location-based clustering schemes. Finally, as a third analytical approach, we compare the performance of our clustering algorithm with the one displayed by another velocity-based clustering protocol introduced in [22]. In this context, the *cluster membership duration* – a parameter directly influencing the cluster lifetime and stability, is considered.

In mathematical terms, vehicles moving on the highway and grouping into cluster structures have normally distributed velocities. The bell-shaped curve of this normal distribution depends on several traffic conditions, such as the type of the highway (local, national or international), the region nearby which the highway passes (a heavy industrialized city or a small village), the time of the day (morning, daytime, evening or night), and so on. In this respect, the well known distribution function takes the form:

$$f(v_{avg(node)}) = \frac{1}{v_{rel}\sqrt{2\pi}} \cdot e^{-\frac{(v_{avg(node)} - v_{avg(cluster)})^2}{2v_{rel}^2}} \tag{6.1}$$

where the standard deviation is seen as the relative velocity (v_{rel}) differentiating the mean ($v_{avg(cluster)}$) from the instant value ($v_{avg(node)}$). In the context of velocity threshold condition, nodes with speeds comprised within the range $[-v_{thresh}, \ldots, v_{thresh}]$ can be grouped into clusters. The $-v_{thresh}$ limit (lower bound) corresponds to nodes moving a bit slower compared to the cluster average velocity, whereas the positive v_{thresh} limit (upper bound) refers to nodes moving slightly faster than the cluster entity as a whole. Nevertheless, if the velocity threshold condition is met, both groups of nodes should act as stable members for a quite long period in terms of cluster lifetime. Figure 6.1 gives a graphical representation to (6.1) and presents an example of a CM node satisfying the velocity threshold condition and consequently participating in the cluster formation.

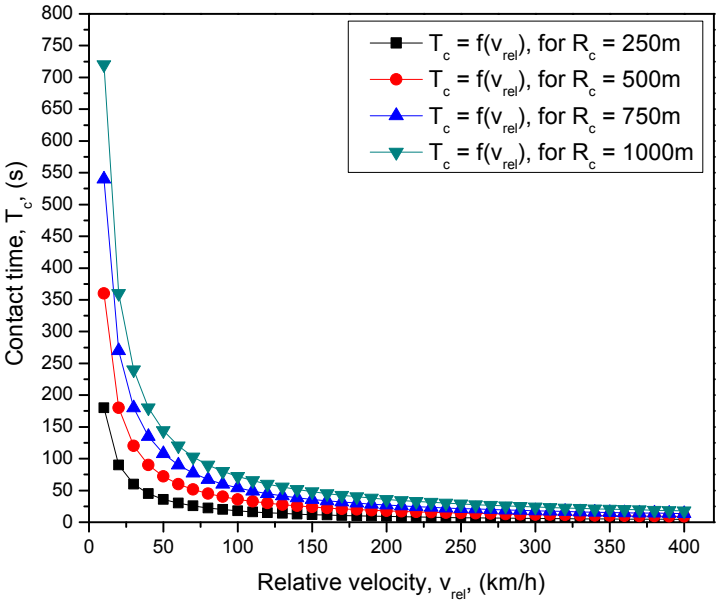


Figure 6.2.: Dependence of the inter-vehicle contact time on the relative velocity and communication range.

6.1.1. Inter-vehicle Contact Time

Inter-vehicle communication in VANETs is organized based on wireless connections established between nodes, where vehicles form communication circles with transmission radii comprised between $250\,m \div 1000\,m$ limits. Of course, the radius length depends on the emitted power of the TX radio antennas, which the nodes are equipped with.

As it was introduced in Chapter 5, the *relative average velocity* is used as CBAP's clustering metric. The lower the relative velocity is, the longer time vehicles stay in mutual contact. This in turn results in more stable cluster structures. With all that said, it is possible to determine the interdependence function of the inter-vehicle contact time (interval during which nodes can sense each other) on the relative velocity between nodes (Figure 6.2). According to this function, a value for the *velocity threshold condition* parameter can easily be chosen. However, it is important to keep in mind that too low threshold values result in small numbers of condition-satisfying nodes, fact which consequently leads to smaller cluster structures. As it was expected, lower relative velocities and higher communication radii result in longer contact times. From the obtained function $T_c = f(v_{rel}, R_c)$, v_{thresh} was chosen equal to 20 km/h. Thus, only nodes with relative average velocities: $v_{rel} \leq 20\,km/h$ can join the cluster. This value for the threshold results in a quite long contact time in terms of cluster lifetime.

Beside the communication range, one of the main parameters determining the cluster lifetime is the rate/probability with which nodes join/leave a cluster. It is obvious that the Poisson distribution, which characterizes the discrete nature of joining/leaving nodes, is

the best model that can be applied in such sort of analysis. In the following evaluation, this Poissonian assumption is further developed.

6.1.2. Cluster Stability

In order to theoretically estimate the performance of the CBAP velocity-based clustering algorithm, a comparison with other types of cluster formation mechanisms can be very helpful and informative with respect to protocol evaluation. In this context, a thorough comparative analysis of the velocity-based clustering versus the location-based clustering is considered. The main difference between the two cluster formation mechanisms is:

1. In velocity-based clustering, only nodes moving in the mutual communication radius and simultaneously satisfying the velocity threshold condition, i.e. having close or similar speeds, can form a cluster structure;
2. In location-based clustering, all nodes moving in the mutual communication range are clustered independent of their velocity.

As it was stated in the previous section, the nodes' velocities follow a normal distribution, characterized by the mean μ and the standard deviation σ values. In terms of velocity-based clustering, the mean/expectation is seen as the cluster average velocity ($\mu = v_{avg(cluster)}$), where different nodes in the CH's communication radius can have a large range of velocity values (characterized by the standard deviation σ) varying around the central mean. Three different traffic scenarios are considered in the following:

1. *low traffic density* ($\mu = 90\,km/h, \sigma = 40\,km/h$);
2. *average traffic density* ($\mu = 70\,km/h, \sigma = 30\,km/h$);
3. *high traffic density* ($\mu = 20\,km/h, \sigma = 10\,km/h$).

In the low traffic density situation, nodes move quite fast and there is a relatively large variation of their velocities on the road. The average traffic density scenario is characterized by lower nodes' speeds and a lower variation of their dynamic characteristics. Finally, specific to high traffic density case (such as a slowly moving traffic jam) is the low node average velocity and a lower deviation from this mean, which results in a more homogeneous distribution of speeds.

In location-based clustering however, nodes can be considered normally distributed in accordance with their geographic coordinates, where clusters are built only taking into account the communication range, regardless of nodes' velocities. In this context, the difference between the two mechanisms comes at the point of considering the nodes' mobility: in velocity-based clustering only nodes with velocities located in the range

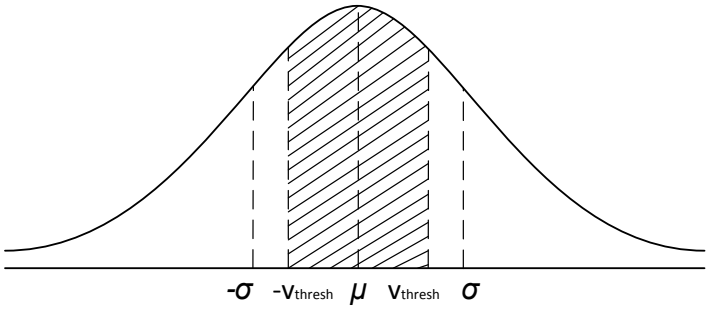


Figure 6.3.: Normal distribution of node velocities in the average traffic density scenario.

$[-v_{thresh}, \ldots, v_{thresh}]$ can become CMs (marked region in Figure 6.3), whereas in location-based clustering velocities are not regarded as decision parameters. On the other hand, the similarity between the two clustering schemes consists in the necessity of mutual sensing among nodes, in order for them to participate in cluster formation. Also, characteristic to both clustering mechanisms, nodes enter the CH transmission range in a Poisson distributed manner, characterized by the rate λ [35]:

$$P(N, \lambda) = e^{-\lambda} \sum_{i=0}^{\lfloor N \rfloor} \frac{\lambda^i}{i!} \tag{6.2}$$

where: λ is the average rate with which nodes enter into the CH communication range (cluster joining rate), and $P(N, \lambda)$ is the Cumulative Distribution Function (CDF) which shows the probability that N or fewer nodes join the cluster within a certain time interval. Note that N is an indicator of cluster stability: the fewer nodes join and leave the cluster in a certain time span, the more stable is the cluster structure.

Because in our velocity-based clustering only nodes that satisfy the velocity threshold condition can become CMs, the joining rate λ is characterized by a special *mobility factor* L and the *cluster joining probability* k:

$$\lambda = k \cdot L. \tag{6.3}$$

The k parameter shows the percentage of joining the cluster nodes, out of the total number of sensed vehicles. For location-based clustering $k = 1$ because all sensed nodes become part of the cluster, hence: $\lambda = L$. For the CBAP velocity-based clustering on the other hand, the factor k expresses the likelihood that nodes in the communication range satisfy the velocity threshold condition and subsequently join the cluster. The values taken by k vary in this case based on the mentioned traffic scenarios, where the velocity distributions are characterized by three different probability density functions (one of them presented in Figure 6.3). The numerical value of the k parameter can be calculated as the probability of successfully joining the cluster nodes. Therefore, based on the aforementioned traffic

Table 6.1.: Parameter Summary

Protocol	Traffic situation	v_{thresh}	μ	σ	k	L
Location-based	low density	–	90	40	1	4
	medium density	–	70	30	1	4
	high density	–	20	10	1	4
Velocity-based	low density	20	90	40	0.38	4
	medium density	20	70	30	0.49	4
	high density	20	20	10	0.95	4

density scenarios and the corresponding values for μ and σ, the general expression for k takes the form:

$$k = \int_{\mu - v_{thresh}}^{\mu + v_{thresh}} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx \tag{6.4}$$

which leads to three k values:

1. low traffic density: $k = 0.38$;
2. average traffic density: $k = 0.49$;
3. high traffic density: $k = 0.95$.

The L mobility factor is independent of the clustering mechanism and depends on the velocity and communication range. As a calculation example, L was chosen equal to: $L = 4$, however the relative differences between the shapes of the Poisson distributions will be the same for different values of L. All the assumed and calculated parameters are summarized in Table 6.1.

Based on the obtained k and L parameters, the cumulative distribution functions dependent on the number of connecting nodes: $P = f(N, \lambda)$ were plotted. As a result, we obtained four graphs, of which three for velocity-based clustering $\lambda_{low} = 0.38 \times L$, $\lambda_{avg} = 0.49 \times L$, $\lambda_{high} = 0.95 \times L$ and one dependence for the location-based clustering: $\lambda_{l.b.c.} = L$ (as shown in Figure 6.4). Analyzing the obtained graphs, it can be seen that velocity-based clustering performs much better in terms of cluster stability than its location-based counterpart. This performance is mainly observed in the low and average traffic density scenarios, where the CDF functions have a steeper slope. This means that it is significantly more likely that fewer nodes join and leave the velocity-based cluster in a certain time interval, fact which results in fewer membership fluctuations – phenomenon which directly affects the cluster stability. On the other hand, as it was expected, in case of high traffic density such as a traffic jam, the two clustering schemes perform quite similar in terms of cluster stability. This is predominantly caused by the fact that in heavy traffic

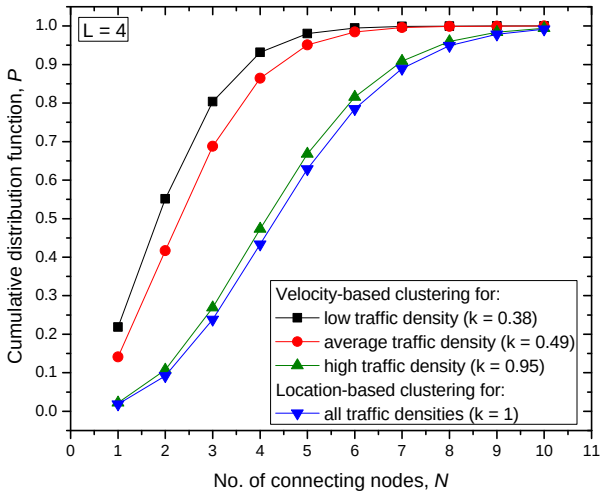


Figure 6.4.: Cumulative distribution functions of the two clustering mechanisms for different traffic scenarios ($L = 4$).

conditions, nodes have both close coordinates/location and similar or very close velocities. If so, almost all vehicles in the communication range satisfy the velocity threshold condition and become part of the cluster. At this point an important conclusion can be drawn: velocity-based clustering is especially promising for traffic situations where velocities are heterogeneous.

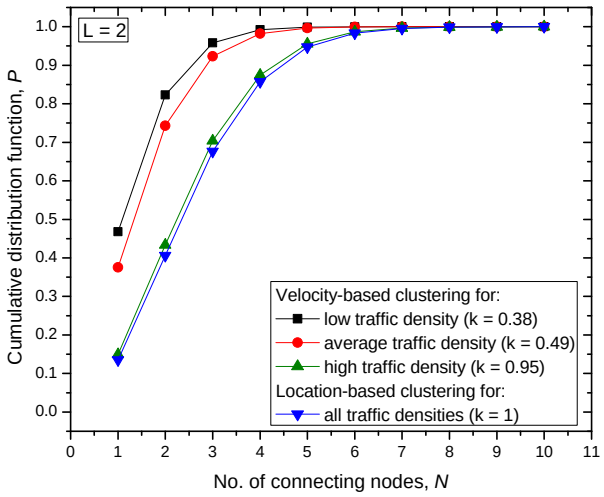
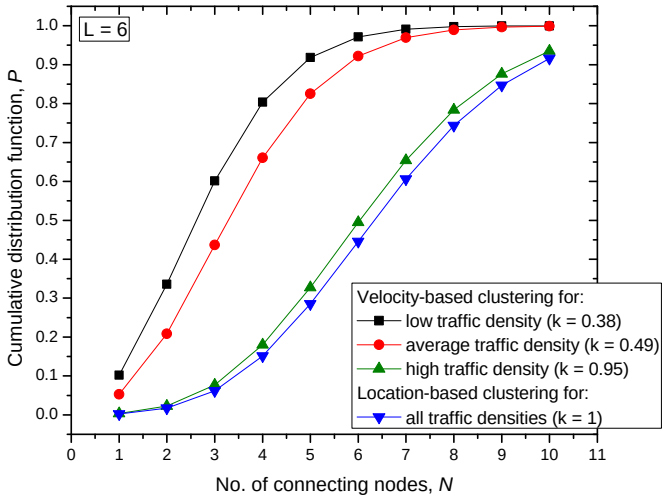


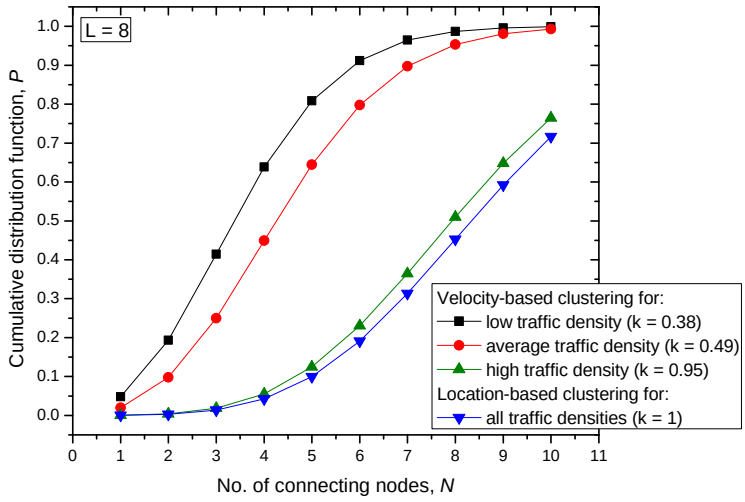
Figure 6.5.: Cumulative distribution functions of the two clustering mechanisms for different traffic scenarios ($L = 2$).

As discussed earlier, there is a direct proportionality between the mobility factor L and the communication radius R_c. In order to graphically represent how the transmission range influences cluster stability, besides the previously plotted graph ($L = 4$), we also

present the CDFs for: $L = 2, 6$ and 8 (Figures 6.5 and 6.6). These four linearly increasing mobility factors are linked to the four earlier mentioned typical communication radii used in IVC. As expected from the analysis of the four obtained CDFs, with an increasing communication radius the probability that more nodes join/leave the cluster increases as well – phenomenon that adversely affects the cluster stability. As a concluding point it should be emphasized that shorter transmission ranges result in higher cluster stability, while larger communication radii result in more fluctuations in the cluster structure. This is one of the basic arguments why we opted for the shortest transmission range while designing our CBAP protocol.



(a) $L = 6$



(b) $L = 8$

Figure 6.6.: Cumulative distribution functions of the two clustering mechanisms for different traffic scenarios.

6.1.3. Cluster Membership Duration

As it was stated and analyzed in Section 5.1.2 of the previous chapter, our clustering algorithm builds clusters based on the node average velocity displayed by each moving vehicle, instead of taking the node instant velocity, as is the case of [22]. We believe that such an approach used in dynamic characteristics consideration is especially valuable in the annihilation of velocity fluctuations encountered during short termed dynamic maneuvers. As it could be seen from Figure 5.3, velocity fluctuations were much smaller in our clustering procedure, than in the other velocity-based counterpart.

In the context of the aforementioned, it makes sense to compare the duration of cluster membership that a node displays in both clustering algorithms, during longer termed accelerations. In this regard, we use the well known relations describing the node's dynamic properties. The node instant velocity is characterized by:

$$v = v_0 + a \times t, \tag{6.5}$$

where: v is the instant velocity after t seconds of node acceleration/deceleration a, and v_0 is the initial instant velocity before acceleration is applied.

A node is preserving its cluster membership as long as $v - v_0 \leq v_{thresh}$, thus the cluster membership duration (T_{MD}) can be calculated as:

$$T_{MD} = \frac{v_{thresh}}{a} = \frac{v_{thresh}}{\Delta v/t} = \frac{v_{thresh}}{v - v_0} \times t, \tag{6.6}$$

where $t = T_{rand} = 1\,s$, and $v_{thresh} = 20\,km/h$. As a result, the calculation of cluster membership duration is based on the difference of previous and instant velocity values. Because we consider further the traffic scenario initiated in Section 5.1.2, $v_0 = 70\,km/h$.

The calculation procedure described till now is basically valid for the clustering algorithm that builds clusters based on the instant velocities, rather than average speeds. In order to obtain equivalent results for our clustering algorithm, we adjust (5.1) to:

$$v_{avg_k} = 0.5 \times v + 0.5 \times v_{avg_{k-1}}, \tag{6.7}$$

where v_{avg_k} is the last node average velocity used in clustering decision, v as before is the instant velocity value after acceleration is applied, and $v_{avg_{k-1}}$ is the previous node average velocity. For the most first v_{avg} calculation, $k = 1$ and $v_{avg_{k-1}} = v_0$. Important is to emphasize that in this case, the T_{MD} value is equal to the number of T_{rand} intervals during which the condition $v_{avg_k} \leq v_{thresh}$ is satisfied.

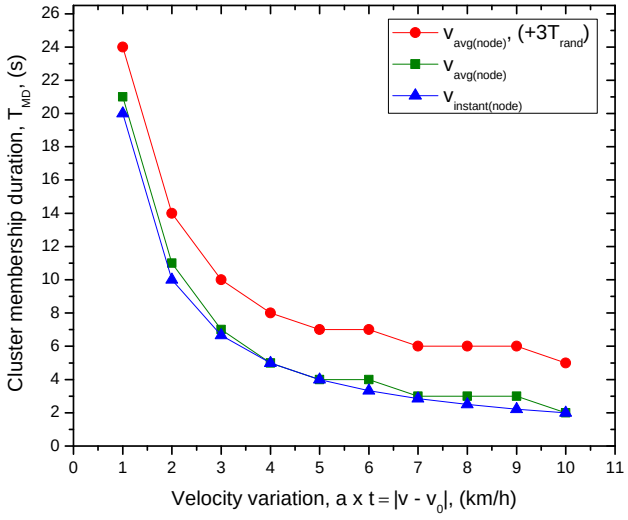


Figure 6.7.: Dependence of cluster membership duration on the node acceleration (velocity variation).

After a few computations, it was possible to plot the dependence $T_{MD} = f(\Delta v)$ presented in Figure 6.7. As it can be seen, the cluster membership duration specific to our clustering mechanism is slightly higher than the one characteristic to [22]. This result can be enhanced by varying the α coefficient, thus giving a higher priority to previous average velocities. Nevertheless, in these circumstances, we improve the cluster membership duration by simply adding a $3 \times T_{rand}$ interval, which results in the red curve characteristic.

As a concluding point it is worth mentioning that for low traffic densities, when a cluster contains only two members, the approximation: $T_{cluster.lifetime} \approx T_{MD}$ becomes valid. Last but not least, it can happen that during the first seconds of acceleration, the CM node simply leaves the CH's communication range R_c. If this is the case, after expiration of a $3 \times T_{rand}$ interval, the cluster membership duration equals: $T_{MD} = 0\,s$.

6.2. Hierarchical Aggregation Scheme Evaluation

In this section we perform a further analysis of the hierarchical aggregation scheme introduced in Section 5.3 of the previous chapter. We analyze the application of the earlier mentioned Bloom filters in the data duplicates detection, as well as introduce a novel approach in data truncation procedure, intended to improve the aggregation mechanism in terms of bandwidth consumption. Last but not least, a parameters' optimization analysis is further presented.

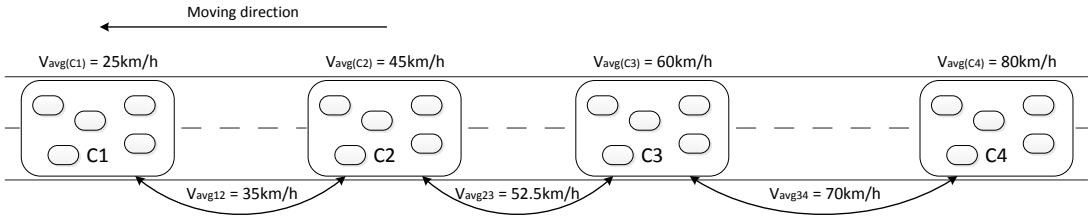


Figure 6.8.: Hierarchical aggregation case scenario.

6.2.1. DDA Case Studies

In order to better understand how the hierarchical aggregation scheme performs on concrete scenarios, we introduce two case studies in the following analysis.

Let us assume 3 adjacent clusters (C2, C3 and C4), driving in the same direction and mutual proximity, are approaching a traffic jam. The three clusters are characterized by cluster average velocities equal to 45 km/h, 60 km/h and 80 km/h respectively, while the traffic congestion is represented by a fourth moving cluster (C1), driving in the most front with a 25 km/h speed (Figure 6.8). The threshold value for data fusion when aggregation is applied is set to $v_{thresh} = 20\,km/h$.

Assume all four clusters have generated different random Cluster IDs, which we arbitrarily set to:

1. CID1 = 20524, binary translated to: 0 1 0 1 0 0 0 0 0 0 1 0 1 1 0 0
2. CID2 = 13346, binary translated to: 0 0 1 1 0 1 0 0 0 0 1 0 0 0 1 0
3. CID3 = 45096, binary translated to: 1 0 1 1 0 0 0 0 0 0 1 0 1 0 0 0
4. CID4 = 02371, binary translated to: 0 0 0 0 1 0 0 1 0 1 0 0 0 0 1 1

In the first case scenario, we assume, C2 and C3 start mutually sensing each other first. They exchange their aggregates (cluster average velocities) $v_{avg(C2)}$ and $v_{avg(C3)}$, check the threshold condition and because it is satisfied: $|45 - 60| \leq 20$, both CHs check the two Cluster IDs against the duplicates presence:

CID2 = 0 0 1 1 0 1 0 0 0 0 1 0 0 0 1 0
CID3 = 1 0 1 1 0 0 0 0 0 0 1 0 1 0 0 0
CCID23 = 1 0 1 1 0 1 0 0 0 0 1 0 1 0 1 0

$X^* = \frac{-16 \times \ln(1-\frac{5}{16})}{5} = 1.19 \approx 1; \quad Y^* = \frac{-16 \times \ln(1-\frac{5}{16})}{5} = 1.19 \approx 1;$

$X^* \cup Y^* = \frac{-16 \times \ln(1-\frac{7}{16})}{5} = 1.84 \approx 2;$

As a result: $X^* \cap Y^* = 1 + 1 - 2 = 0 < 1 \Longrightarrow$ no duplicates found.

As the Duplicate Detection Algorithm issues a result of "0 identified duplicates", both in-network aggregates are hierarchically aggregated. Meanwhile, C2 starts sensing C1 and C3 notices C4 in its neighborhood. Because of the previous hierarchical aggregation, C2 and C3 start disseminating an identical hierarchical aggregate $v_{avg23} = 52.5\,km/h$, which is received by clusters C3 and C4. Consequently, this aggregate is then compared between C3 and C4 with aggregate $v_{avg(C4)} = 80\,km/h$, respectively between C1 and C2, with aggregate $v_{avg(C1)} = 25\,km/h$. Because both differences: $|v_{avg23} - v_{avg1}| = |52.5 - 25| > 20$ and $|v_{avg23} - v_{avg4}| = |52.5 - 80| > 20$ do not meet the threshold condition, the aggregates are not further hierarchically aggregated, instead are kept separate and disseminated until all clusters get 3 different aggregated views: $25\,km/h$, $52.5\,km/h$ and $80\,km/h$. Hereby, the network converges. For clusters C3 and C4 these data tend to be very informative, since both get acquainted that further on the road a dramatic decrease of speed follows (conclusion: accident or traffic jam).

As a second case scenario assume C2 senses at the same time both C1 and C3, while C3 simultaneously senses C2 and C4. As a first step, the in-network aggregates are checked, whether they satisfy the threshold condition:

1. C2 checks: $|v_{avg(C1)} - v_{avg(C2)}| < 20$ and $|v_{avg(C2)} - v_{avg(C3)}| < 20$, and
2. C3 checks: $|v_{avg(C2)} - v_{avg(C3)}| < 20$ and $|v_{avg(C3)} - v_{avg(C4)}| < 20$.

As it can be seen from Figure 6.8, all four comparisons are satisfying the threshold condition, hence the Duplicate Detection Algorithm is applied. Following the same procedure as in the first case scenario, three pairs of checks are executed now (for more details, see Appendix A):

1. CID1-CID2 DDA Check, performed by both CH1 and CH2,
2. CID2-CID3 DDA Check, performed by both CH2 and CH3, and
3. CID3-CID4 DDA Check, performed by both CH3 and CH4.

After a few simple calculations, all three DDA checks have resulted in no overlaps of the Cluster IDs associated with the four aggregates, meaning none of the Cluster IDs repeats in the adjacent Cluster IDs. As a result, the hierarchical aggregation of all three pairs of aggregates can be performed, operation which leads to 3 hierarchical aggregates, namely:

$$v_{avg12} = 35\,km/h, v_{avg23} = 52.5\,km/h \; and \; v_{avg34} = 70\,km/h. \tag{6.8}$$

Important is to emphasize that all three resulted hierarchical aggregates are further linked with the joint CCIDs calculated according to the aforementioned scheme. As a result, cluster C1 holds one single aggregate: $v_{avg12} = 35\,km/h$, C2 contains 2 aggregates:

$v_{avg12} = 35\,km/h$ and $v_{avg23} = 52.5\,km/h$, C3 possesses aggregates: $v_{avg23} = 52.5\,km/h$ and $v_{avg34} = 70\,km/h$, and finally C4 holds the single aggregate: $v_{avg34} = 70\,km/h$. From the resulted hierarchical aggregates it can be easily seen that both tuples v_{avg12} and v_{avg23} contain a duplicate v_2, as well as aggregates: v_{avg23} and v_{avg34} include the duplicate v_3. Of course this is a duplicate issue, that has to be identified in order to prevent the sequential hierarchical aggregation.

Further on, when aggregates v_{avg12} and v_{avg23} are compared by CH2, the threshold condition is fulfilled: $|35 - 52.5| < 20$, just as in the comparison case of v_{avg23} and v_{avg34} performed by CH3: $|52.5 - 70| < 20$. As a result, the Duplicate Detection Algorithm is again applied:

1. DDA applied by CH2:

   CCID12 = 0 1 1 1 0 1 0 0 0 0 1 0 1 1 1 0

   CCID23 = 1 0 1 1 0 1 0 0 0 0 1 0 1 0 1 0

   CCID1223 = 1 1 1 1 0 1 0 0 0 0 1 0 1 1 1 0

   $X^* = \frac{-16\times\ln(1-\frac{8}{16})}{5} = 2.21 \approx 2;\quad Y^* = \frac{-16\times\ln(1-\frac{7}{16})}{5} = 1.84 \approx 2;$

   $X^* \cup Y^* = \frac{-16\times\ln(1-\frac{9}{16})}{5} = 2.64 \approx 3;$

   As a result: $X^* \cap Y^* = 2 + 2 - 3 = 1$ duplicate found $\Longrightarrow$ do not allow hierarchical aggregation $\Longrightarrow$ apply the second checking stage:

   CCID12 = 0 1 1 1 0 1 0 0 0 0 1 0 1 1 1 0

   CCID23 = 1 0 1 1 0 1 0 0 0 0 1 0 1 0 1 0

   AND = 0 0 1 1 0 1 0 0 0 0 1 0 1 0 1 0 $\neq$ CCID12, CCID23 (no match found)

   $\Longrightarrow$ disseminate both aggregates.

2. DDA applied by CH3:

   CCID23 = 1 0 1 1 0 1 0 0 0 0 1 0 1 0 1 0

   CCID34 = 1 0 1 1 1 0 0 1 0 1 1 0 1 0 1 1

   CCID2334 = 1 0 1 1 1 1 0 1 0 1 1 0 1 0 1 1

   $X^* = \frac{-16\times\ln(1-\frac{7}{16})}{5} = 1.84 \approx 2;\quad Y^* = \frac{-16\times\ln(1-\frac{10}{16})}{5} = 3.13 \approx 3;$

   $X^* \cup Y^* = \frac{-16\times\ln(1-\frac{11}{16})}{5} = 3.72 \approx 4;$

   As a result: $X^* \cap Y^* = 2 + 3 - 4 = 1$ duplicate found $\Longrightarrow$ do not allow hierarchical aggregation $\Longrightarrow$ apply the second checking stage:

   CCID23 = 1 0 1 1 0 1 0 0 0 0 1 0 1 0 1 0

   CCID34 = 1 0 1 1 1 0 0 1 0 1 1 0 1 0 1 1

   AND = 1 0 1 1 0 0 0 0 0 0 1 0 1 0 1 0 $\neq$ CCID23, CCID34 (no match found)

   $\Longrightarrow$ disseminate both aggregates.

From the obtained results, it can be noticed that despite both pairs of aggregates held by CH2 and CH3 satisfy the threshold condition, the DDA is not fulfilled, hence no further hierarchical aggregation is applied. In this context, each CH disseminates both aggregates: CCID12 and CCID23 are disseminated to clusters C1 and C3, and CCID23 and CCID34 are disseminated to clusters C2 and C4. As a result, CH3 gets acquainted with upcoming situation represented by hierarchical aggregate CCID12 and C4 is informed about the same upcoming decrease in speed displayed by CCID23. This aggregates exchange among clusters results in:

1. CH1 holding: CCID12, with $v_{avg12} = 35\,km/h$ and CCID23, with $v_{avg23} = 52.5\,km/h$;
2. CH2 holding: CCID12, with $v_{avg12} = 35\,km/h$, CCID23, with $v_{avg23} = 52.5\,km/h$ and CCID34, with $v_{avg34} = 70\,km/h$;
3. CH3 holding: CCID12, with $v_{avg12} = 35\,km/h$, CCID23, with $v_{avg23} = 52.5\,km/h$ and CCID34, with $v_{avg34} = 70\,km/h$;
4. CH4 holding: CCID23, with $v_{avg23} = 52.5\,km/h$ and CCID34, with $v_{avg34} = 70\,km/h$.

Once all four CHs hold multiple aggregates, they start checking whether the newly received tuples can be hierarchically aggregated with the local ones. Thus:

- CH1 compares: $|v_{avg12} - v_{avg23}| = |35 - 52.5| < 20 \Rightarrow$ threshold condition is satisfied, but the duplicate detection algorithm detects a repeating aggregate (CID2) $\Rightarrow$ disseminate both aggregates separately;
- CH2 knows from the previous analysis that v_{avg12} and v_{avg23} are incompatible for hierarchical aggregation, therefore compares only:
  1. $|v_{avg23} - v_{avg34}| = |52.5 - 70| < 20 \Rightarrow$ threshold condition is fulfilled, but the DDA returns a result of one repeating aggregate (CID3) $\Rightarrow$ disseminate both aggregates separately;
  2. $|v_{avg12} - v_{avg34}| = |35 - 70| > 20 \Rightarrow$ threshold condition not satisfied $\Rightarrow$ disseminate both aggregates separately.
- CH3 knows from the previous aggregate analysis that v_{avg23} and v_{avg34} are disparate for hierarchical aggregation, hence compares only:
  1. $|v_{avg12} - v_{avg23}| = |35 - 52.5| < 20 \Rightarrow$ threshold condition is met, but the DDA returns a result of one repeating tuple (CID2) $\Rightarrow$ disseminate both aggregates separately;;
  2. $|v_{avg12} - v_{avg34}| = |35 - 70| > 20 \Rightarrow$ threshold condition not satisfied $\Rightarrow$ disseminate both aggregates separately.

- CH4 compares: $|v_{avg23} - v_{avg34}| = |52.5 - 70| < 20 \Rightarrow$ threshold condition is satisfied, but the DDA detects a repeating aggregate (CID3) $\Rightarrow$ disseminate both aggregates separately.

6.2.2. Network Convergence

After performing the aforementioned analysis, CH1 starts disseminating 2 aggregates (CCID12 and CCID23), CH2 and CH3 disseminate 3 aggregates (CCID12, CCID23 and CCID34), and finally CH4 disseminates 2 aggregates (CCID23 and CCID34). As a result, C4 finds out that a further decrease in velocity follows: $70km/h \longrightarrow 52.5km/h \longrightarrow 35km/h$. Due to the fact that inter-cluster communication is performed exclusively via CMGW nodes, it takes an interval of exactly $2 \times T_{rand}$ for one mutual aggregate exchange between 2 CHs. For this particular case, C4 gets informed about the frontal traffic jam (i.e., receives aggregate v_{avg12}) within exactly $6 \times T_{rand}$ intervals. Consequently, after six consecutively exchanged beacons, the inter-vehicle network converges (all clusters know about each other). From this dissemination scheme an important conclusion can be drawn: a dynamical network converges after $2 \times (N-1)$ exchanges of HELLO packets, where N is the number of adjacent mutually communicating clusters, and $N-1$ is the number of inter-cluster links.

This hierarchical aggregation scheme tends to be quite robust in the sense that for a homogeneous traffic situation, where clusters share similar or close mobility patterns, each CH will be able to hierarchically aggregate the aggregates originated from its adjacent clusters, for instance: CH2 and CH3 will be able to merge CCIDs 12 and 34, resulting in a larger aggregate CCID1234. This will consequently permit a further dissemination of only one aggregate, instead of three as described earlier, since CCID1234 contains the individual in-network aggregates generated by each cluster. The following example shows that the aggregation of CCID12 and CCID34 returns no duplicates detection:

CCID12 = 0 1 1 1 0 1 0 0 0 0 1 0 1 1 1 0
CCID34 = 1 0 1 1 1 0 0 1 0 1 1 0 1 0 1 1
CCID1234 = 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1

$$X^* = \frac{-16 \times \ln(1-\frac{8}{16})}{5} = 2.21 \approx 2; \quad Y^* = \frac{-16 \times \ln(1-\frac{10}{16})}{5} = 3.13 \approx 3;$$

$$X^* \cup Y^* = \frac{-16 \times \ln(1-\frac{13}{16})}{5} = 5.35 \approx 5;$$

As a result: $X^* \cap Y^* = 2 + 3 - 5 = 0 < 1 \Longrightarrow$ no duplicates found $\Longrightarrow$ allow hierarchical aggregation.

The robustness and efficiency of our duplicate detection algorithm is quite easily analyzed and checked using the case scenario, for which a small CCID is fully contained within a larger CCID:

CCID1234 = 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1

CCID23 = 1 0 1 1 0 1 0 0 0 0 1 0 1 0 1 0

CCID122334 = 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1

$X^* = \frac{-16\times\ln(1-\frac{13}{16})}{5} = 5.35 \approx 5;\quad Y^* = \frac{-16\times\ln(1-\frac{7}{16})}{5} = 1.84 \approx 2;$

$X^* \cup Y^* = \frac{-16\times\ln(1-\frac{13}{16})}{5} = 5.35 \approx 5;$

As a result: $X^* \cap Y^* = 5 + 2 - 5 = 2$ duplicates found $\Longrightarrow$ do not allow hierarchical aggregation $\Longrightarrow$ apply the second checking stage:

CCID1234 = 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1

CCID23 = 1 0 1 1 0 1 0 0 0 0 1 0 1 0 1 0

AND = 1 0 1 1 0 1 0 0 0 0 1 0 1 0 1 0 = CCID23 (match found)

$\Longrightarrow$ CCID23 completely contained within CCID1234 $\Longrightarrow$ discard CCID23 and disseminate further only CCID1234.

6.2.3. False Positives Phenomenon

As it was explained in the introductory part of this section, due to the probabilistic nature of Bloom filters, based on which our Duplicate Detection Algorithm works, false positive results can also be obtained. In the context of our hierarchical aggregation scheme, a false positive arises in that case when a CID or CCID is checked against another CCID, in order to identify certain duplicates/common items, and despite of no repeating CIDs, the algorithm returns a positive result, i.e., duplicate(s) found. Just to make the problem clearer, consider the following example. Assume two other adjacent clusters C5 and C6, having CID5 and CID6 are driving in mutual proximity of the aforementioned four clusters. Supposedly, the aggregates v_{avg56} and $v_{avg1234}$ met the threshold condition and the two CCIDs are further analyzed using the DDA. Assume:

CID5 = 0 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0

CID6 = 1 0 1 0 1 0 0 1 0 0 0 0 0 0 1 0

CCID56 = 1 1 1 1 1 0 0 1 0 0 1 1 0 0 1 0

Analyzing further the two CCIDs results in:

CCID1234 = 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1

CCID56 = 1 1 1 1 1 0 0 1 0 0 1 1 0 0 1 0

CCID123456 = 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1

$X^* = \frac{-16\times\ln(1-\frac{13}{16})}{5} = 5.35 \approx 5;\quad Y^* = \frac{-16\times\ln(1-\frac{9}{16})}{5} = 2.64 \approx 3;$

$X^* \cup Y^* = \frac{-16\times\ln(1-\frac{14}{16})}{5} = 6.65 \approx 7;$

As a result: $X^* \cap Y^* = 5 + 3 - 7 = 1$ duplicate found (false positive) $\Longrightarrow$ do not allow hierarchical aggregation $\Longrightarrow$ apply the second checking stage:

CCID1234 = 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 1

CCID56 = 1 1 1 1 1 0 0 1 0 0 1 1 0 0 1 0

AND = 1 1 1 1 1 0 0 1 0 0 1 0 0 0 1 0 $\neq$ CCID1234, CCID56 (no match found)

$\Longrightarrow$ disseminate both aggregates.

As this example shows, despite of none single repeating CID, the DDA issues an erroneous result, that turns to be a false positive. It is important to emphasize in this respect that the false positives come to a cost of extra-bandwidth use, while the unique aggregates are preserved and further disseminated. Basically, the main factor which determines the DDA to issue a false positive is the number of component CIDs combined within the CCID. Thus, the more elements are added to the general CCID, the higher the probability of false positives. In the following we introduce a thorough evaluation of the probability of false positives, analyzed from the perspective of hierarchical aggregation.

The probability of a false positive for an element not being in the set, or also called *false positive rate* can be calculated in a straightforward way. After all the elements of the set S are hashed and added to the Bloom filter, the probability that a specific bit is still 0 is:

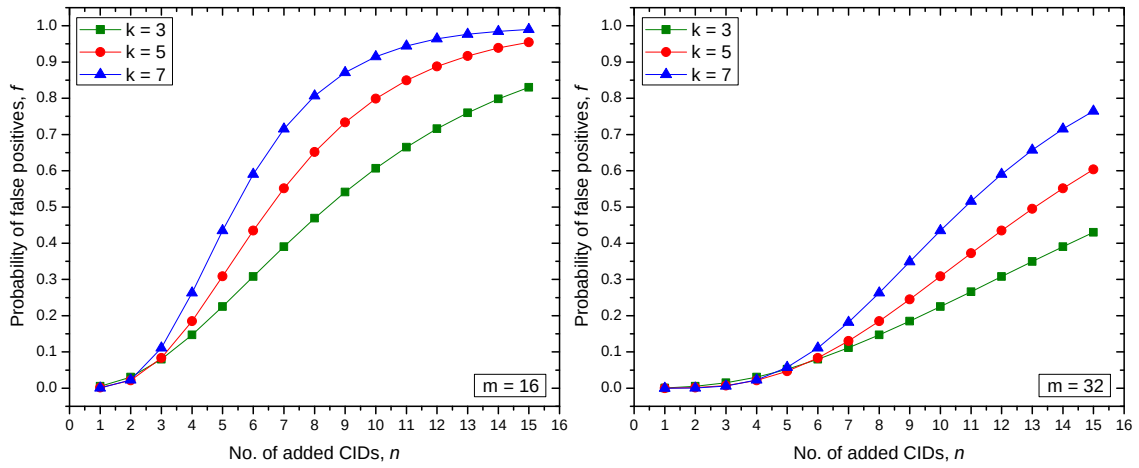
$$p = \left(1 - \frac{1}{m}\right)^{kn} \approx e^{-kn/m} \tag{6.9}$$

where: m is the length of the Bloom filter, k – the number of hash functions, and n – the number of added elements [28].

Let ρ be the proportion of 0 bits after all n elements have been inserted in the Bloom filter. The expected value of ρ in this case is of course: $E(\rho) = p$. In this respect, the probability of a false positive is: $(1 - \rho)^k \approx (1 - p)^k$, or in other words [28]:

$$f = (1 - p)^k = \left(1 - e^{-kn/m}\right)^k. \tag{6.10}$$

As surmised, the probability of false positives increases, when the number of combined CIDs follows an increasing rate as well (common for high densities of adjacent clusters, such as occurring in traffic jams) (Figure 6.9). It is worth mentioning that the probability of false positives (f) also depends on the length of the binary array (m) representing the CID. In this context, shorter the array, higher the probability of false positives (Figure 6.9a), and vice-versa, longer arrays result in lower probabilities of getting a false positive (Figure 6.9b). Nevertheless, this is to be expected, since longer CIDs have a lower changing impact on the resulting final CCID.



(a) $m = 16$ (b) $m = 32$

Figure 6.9.: Probability of false positives (f) as a function of the amount of CIDs combined within the CCID.

6.2.4. Parameter Optimization

Another interesting aspect is the dependence of f on the number of hash functions applied in the mapping of CIDs. Using more hash functions simplifies the process of finding a 0 bit for an element that is not a member of S. On the other hand, using fewer hash functions increases the fraction of 0 bits within the final array. If so, it makes sense to optimize the number of hash functions (k_{opt}), given certain initial parameters, such as m and n.

The optimal number of hash functions that minimizes the false positive rate f is quite easily determined by taking the derivative [28]:

$$\frac{df}{dk} = \left[\left(1 - e^{-kn/m}\right)^k\right]'_k = \left[e^{k \ln\left(1-e^{-kn/m}\right)}\right]'_k = \left[k \ln\left(1 - e^{-kn/m}\right)\right]'_k \times e^{k \ln\left(1-e^{-kn/m}\right)} = 0. \tag{6.11}$$

From the last product, the left part is equalized to 0 and further derived, thus resulting in:

$$\ln\left(1 - e^{-kn/m}\right) = -\frac{kn}{m} \times \frac{e^{-kn/m}}{1 - e^{-kn/m}}. \tag{6.12}$$

On the other hand: $p = e^{-kn/m} \Rightarrow -\frac{kn}{m} = \ln p$, which after substitution in (6.12) returns:

$$\ln(1-p) = \ln p \times \frac{p}{1-p} \Rightarrow 1 - p = p \Rightarrow \boxed{p = \frac{1}{2}}. \tag{6.13}$$

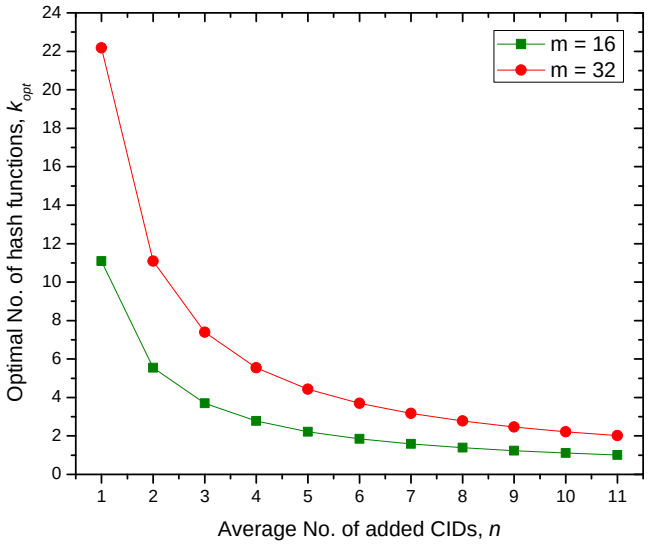


Figure 6.10.: The optimal number of hash functions depending on the average amount of neighboring clusters (CIDs), for two different lengths of the CID array.

Returning back to our initial variables, we calculate k_{opt}:

$$e^{-kn/m} = \frac{1}{2} \Rightarrow e^{kn/m} = 2 \Rightarrow \boxed{k_{opt} = \ln 2 \times \frac{m}{n}}, \tag{6.14}$$

where n shows the average number of CIDs considered for hierarchical aggregation into the CCID (Figure 6.10). In this context, n depends on the clusters' density, which in turn is a function of the road, region and traffic types, driving hours, etc.

In practice, it is obvious that k is an integer, and a smaller, suboptimal k is preferred, because this reduces the number of hash functions that have to be computed. In our particular case implemented in CBAP, we use a fixed value for the number of hash functions ($k = 5 = const$) just to provide an explanatory example of our hierarchical aggregation scheme and its functionality. According to Figure 6.10, as long as the average number of neighboring clusters is low (a low-average traffic density), more hash functions can be applied in the generation of CIDs, and the opposite case (fewer hash functions used) happens when the traffic density increases (rush-hours scenario). This fact suggests a possible conceptual idea: the CID could be generated using a variable number of hash functions, adjusted to the specific traffic scenario, a cluster/CH is part of. In this regard, it is important to understand that in case of a dynamic k implementation, a synchronization between the number of hash functions applied by different clusters is necessary, because a difference in these values can lead to erroneous aggregation results. The synchronization in this case could be based on the statistical dependencies of traffic densities on the driving hours. That is, in the context of a highway scenario, morning and evening hours are

characterized by a lower traffic density than during the day hours (hence, a higher k value can be applied), while in a city scenario it is the other way round. With that said, in our opinion, a dynamic k could improve the efficiency of hierarchical aggregation mechanisms accordingly, however this remains being a matter of future work.

Another solution for increasing the robustness of our hierarchical aggregation scheme, is to use longer Cluster IDs (e.g., $m = 32$). As it was shown in the earlier comparison examples, this could solve/reduce the problem of false positives quite easily, at the expense of 2 additional extra Bytes used for the clustering overhead. In our hierarchical aggregation scheme, Cluster IDs are assumed to have a length of 16 bits, which is a trade-off between the demanding requirements of the DDA threshold (i.e., no duplicates allowed) and a lower amount of bandwidth resources necessary for transmission of a 16 bit-long CCID. On the other hand, the 32 bit-long alternative can be applied when lower demands are set for DDA, i.e., few duplicates are allowed. In this situation, longer CCIDs can be tolerated, since in most of the cases hierarchical aggregation will result in only one single aggregate to be disseminated (even if it contains a few duplicates).

Last but not least, the Time-To-Live (TTL) timer (analyzed in Subsection 5.3.5) is also intended to prevent a too long lifetime of HELLO beacons carrying the aggregates, that become irrelevant over time. This is specifically beneficial for the hierarchical aggregation, since a packet dies out before too many CIDs are fused and a false positive might arise.

6.2.5. Aggregates Truncation

If a CH node holds multiple aggregates that for different reasons (unsatisfied threshold condition or DDA) are not hierarchically aggregated, the resulting tuples are kept separate in the Neighbor Cluster Table. As a result, the CH disseminates all these data structures in order to provide its cluster member nodes, as well as adjacent clusters with most precise and accurate information regarding the real traffic situation. In the context of data aggregation, it becomes crucial to estimate the maximum admissible margin, or the approximate maximum number of aggregates, that a beacon can carry at a single data dissemination. This is particularly important to consider, while trying to make our CBAP protocol more feasible and robust in terms of bandwidth consumption.

In this individual case, the best solution is to apply data truncation, or in other words, choose the most valuable aggregates and disseminate them at a single data exchange. If the amount of aggregates contained in the neighbor cluster table is below the aforementioned admissible margin, all data tuples can be included in one single beacon and subsequently transmitted. However, if this amount exceeds the preset threshold, according

to CBAP, only M aggregates with the highest values set for TTL will be considered for further dissemination. In this context, TTL is used as a decision parameter, because a higher TTL first of all means a lower velocity characterizing the aggregate, and secondly, such an aggregate reaches longer distances while still being relevant.

In order to determine the maximum admissible margin (M), it is important to accurately consider the main parameters carried by the clustering overhead, while taking into account the maximum amount of information each header's field can contain. Assume a maximum transmission unit (MTU) of 1500 Bytes, which can be disseminated at most at one single transmission session. The MTU packet structure contains:

1. a CAM beacon, which plays the role of the main data carrier whose length: $L_{CAM} = 800\,Bytes$ [32], [33];
2. an optional security header, used for data encryption, authentication and authorization purposes. It has a variable length, depending on the security algorithms and applications involved, sometimes including also data certificates. As a result the security header can reach up to $H_{sec} = 250, ..., 300\,Bytes$ [34];
3. and finally, our clustering overhead ($H_{cluster}$) piggybacked to the CAM message.

While making the calculations, it is also important to take the MTU a bit smaller than its maximum value, since longer data structures are more prone to transmission-related errors. For a simple, demonstrative calculus, we consider the effective MTU size equal to: $MTU_{eff} = 1400\,Bytes$. As a result:

$$H_{cluster} = MTU_{eff} - L_{CAM} - H_{sec} \approx 1400 - 800 - 300 = 300\,Bytes. \tag{6.15}$$

Based on the knowledge that each aggregate is a quite complex data structure, containing such fields as: (Combined) Cluster ID, Geo-location-descriptor, Cluster Average Velocity, TTL, Timestamp, Number of Nodes in the cluster, etc, and also taking into account the approximative HELLO beacon's parameters presented in Figure 5.4, we assume an aggregate length of approximately:

$$L_{aggreg} \approx 50\,Bytes \Rightarrow M = \frac{H_{cluster}}{L_{aggreg}} = \frac{300\,Bytes}{50\,Bytes} = 6\,,(aggregates). \tag{6.16}$$

As a result of this calculation, the maximum admissible margin is: $M = 6$. Due to the few approximations made along this computation, we consider taking the admissible margin a bit lower than its maximum value, in order to ensure a degree of reliability and accuracy, and prevent possible errors caused by the probability of having aggregates longer than $L_{aggreg} = 50\,Bytes$. From this consideration, we get: $M_{effect} = 4$.

From the obtained computation results, it has to be concluded that the aggregate truncation algorithm considers only four aggregates with the longest TTL, for further transmission, while the rest are simply discarded. If however the number of too-different aggregates in the neighbor cluster table is ≤ 4, then all data tuples are packed into the HELLO packet's payload and subsequently disseminated.

6.3. Conclusions

In this chapter, we presented a thorough analysis of our velocity-based clustering algorithm implemented in CBAP, while performing an analytical comparison with other types of clustering protocols, namely, the one building clusters based exclusively on vehicles' location characteristics, as well as an analogue velocity-based clustering algorithm introduced in [22]. In this respect, our clustering mechanism proved to be more efficient in terms of cluster stability than its location-based counterpart. It is especially promising in situations where traffic is characterized by a heterogeneous nature of vehicles distribution. On the other hand, compared to its velocity-based counterpart, our clustering algorithm showed a slightly better cluster membership duration (T_{MD}), due to consideration of the node average velocity, instead of the node instant speed. We improve this result further by applying the additional $3 \times T_{rand}$ interval, during which the node preserves its cluster membership (i.e., its node state).

As far as the evaluation of the hierarchical aggregation scheme is concerned, a few case studies showing different aggregation scenarios have been thoroughly parsed. The duplicate detection algorithm proved to be quite efficient in terms of duplicates identification and their consequent resolution. As it was shown, the main limitations related to the probability of false positives and formation of full 1-bit arrays can be easily addressed using longer Cluster IDs, variable or adjustable numbers of hash functions, as well as the earlier introduced TTL field in the clustering overhead.

Talking about reasonable bandwidth consumption, we have presented the *maximum admissible margin* applied in data truncation. It is mainly used when there are too many aggregates that cannot be hierarchically aggregated, but still need to be transmitted. In this situation, TTL proves to be again very helpful, since it is used as a decision criterion parameter/metric.

7

Conclusions and Future Work

In this Master Thesis a novel cluster-based aggregation protocol (CBAP) was proposed and analyzed. The novelty introduced by our protocol relates to the new approach in data aggregation, organized atop prebuilt vehicular cluster structures. Basically, two component sub-protocols are responsible for the CBAP's functionality: a clustering algorithm, organizing vehicles into clusters, and an in-network/hierarchical aggregation scheme dealing with intra- and inter-cluster data aggregation. The main motivation behind this concept was the idea that clustering and aggregation can work as a unitary mechanism, where both schemes benefit from each other. Thus, clustering could be very helpful and supportive in organizing efficient, robust and scalable aggregation schemes by simply ensuring more reliable and longer lasting inter-vehicle connections, while the decision criteria found in aggregation schemes could be applied in formation of more stable and durable clusters.

In order to give a short, and at the same time a thorough overview of the main features, procedures, and functionality aspects designed in this work, the following bullet points will cover the main characteristics of CBAP.

- A novel approach in vehicles' clustering has been introduced, which is mainly based on the similarity degree of nodes' dynamic characteristics. As a result, a new clustering metric has been proposed, namely, the *relative average velocity* between nodes.

- The clustering decision is resembling in many aspects the data aggregation decision criteria, in the sense that only nodes displaying similar or close dynamic characteristics are allowed to form/join a cluster. In this respect, an integration and concatenation of clustering and aggregation mechanisms was pursued, mainly trying to make both sub-protocols work interdependently, while using common parameters and metrics.
- It has been shown that in-network aggregation has a strong impact on the cluster formation and maintenance procedures, where the aggregate computed by the CH node is used in cluster joining and re-clustering decisions. This proves again that both schemes are mutually interrelated.
- Compared to other existing methods for velocity-based clustering, our protocol improves cluster stability by considering the node average velocities, instead of taking the instant speed values into account. Such an approach can be extremely beneficial in mitigation of instability effects caused by short-termed accelerations related to different dynamic maneuvers.
- A novel hierarchical aggregation scheme has also been introduced in this work. Its functionality is based on the operational logic of Bloom filters, where Cluster IDs are regarded as valid Bloom filters and are used in aggregates' identification. These IDs linked to the aggregated tuples are especially valuable in checking data against duplicates presence, and the subsequent duplicates resolution.
- A new timing method related to application of a Time-To-Live parameter has been presented. Its main purpose is to improve the decisions regarding data relevance and aggregates' lifetime.
- Last but not least, different studies related to the false positives presence, parameters optimization and the maximum admissible margin applied in data truncation have been conducted.

From the obtained analytical results it should be concluded that our clustering sub-protocol proves to be an efficient solution in achieving stable clusters, being especially promising in heterogeneous traffic scenarios. As far as the hierarchical aggregation scheme is concerned, the first results show that the adaptation of Bloom filters can be quite robust and effective in duplicates detection. The only issue that might arise in this context are the false positives and the establishment of full 1-bit arrays, however, these problems are easily addressed by means of longer Cluster IDs and utilization of fewer hash functions.

As future work, more simulation analyses will be performed, while trying to improve the hierarchical aggregation scheme by using a variable number of hash functions, Cluster IDs of different lengths, and other parameters optimizations.

A

Appendix

In the second study case scenario introduced in Subsection 5.3.4, after each cluster exchanges aggregates with its adjacent clusters, all CH nodes hold multiple data tuples. After the velocity threshold condition is checked and subsequently satisfied, the Duplicate Detection Algorithm comes into action. As a result, the following Cluster IDs are further analyzed against duplicates presence:

1. CID1-CID2 DDA Check, performed by both CH1 and CH2:

   CID1 = 0 1 0 1 0 0 0 0 0 0 1 0 1 1 0 0

   CID2 = 0 0 1 1 0 1 0 0 0 0 1 0 0 0 1 0

   CCID12 = 0 1 1 1 0 1 0 0 0 0 1 0 1 1 1 0

   $X^* = \frac{-16\times\ln(1-\frac{5}{16})}{5} = 1.19 \approx 1;\quad Y^* = \frac{-16\times\ln(1-\frac{5}{16})}{5} = 1.19 \approx 1;$

   $X^* \cup Y^* = \frac{-16\times\ln(1-\frac{8}{16})}{5} = 2.21 \approx 2;$

   As a result: $X^* \cap Y^* = 1+1-2 = 0 < 1 \Longrightarrow$ no duplicates found $\Longrightarrow$ allow hierarchical aggregation.

2. CID2-CID3 DDA Check, performed by both CH2 and CH3:

   CID2 = 0 0 1 1 0 1 0 0 0 0 1 0 0 0 1 0

   CID3 = 1 0 1 1 0 0 0 0 0 0 1 0 1 0 0 0

   CCID23 = 1 0 1 1 0 1 0 0 0 0 1 0 1 0 1 0

$X^* = \frac{-16\times\ln(1-\frac{5}{16})}{5} = 1.19 \approx 1;\quad Y^* = \frac{-16\times\ln(1-\frac{5}{16})}{5} = 1.19 \approx 1;$

$X^* \cup Y^* = \frac{-16\times\ln(1-\frac{7}{16})}{5} = 1.84 \approx 2;$

As a result: $X^* \cap Y^* = 1+1-2 = 0 < 1 \Longrightarrow$ no duplicates found $\Longrightarrow$ allow hierarchical aggregation.

3. CID3-CID4 DDA Check, performed by both CH3 and CH4:

   CID3 = 1 0 1 1 0 0 0 0 0 0 1 0 1 0 0 0

   CID4 = 0 0 0 0 1 0 0 1 0 1 0 0 0 0 1 1

   CCID34 = 1 0 1 1 1 0 0 1 0 1 1 0 1 0 1 1

$X^* = \frac{-16\times\ln(1-\frac{5}{16})}{5} = 1.19 \approx 1;\quad Y^* = \frac{-16\times\ln(1-\frac{5}{16})}{5} = 1.19 \approx 1;$

$X^* \cup Y^* = \frac{-16\times\ln(1-\frac{10}{16})}{5} = 3.13 \approx 3;$

As a result: $X^* \cap Y^* = 1 + 1 - 3 = -1 < 1 \Longrightarrow$ no duplicates found $\Longrightarrow$ allow hierarchical aggregation.

List of Figures

1.1. Intelligent Transport Systems [3]. 2
1.2. Example of a VANET topology in a city-traffic scenario [6]. 4
1.3. A VANET node's onboard equipment (based on [7]). 6
1.4. Aggregation vs. Non-aggregation Schemes (based on [10]). 9
1.5. Architectural model of an aggregation scheme [10]. 10

4.1. The common metrics used in clustering and aggregation decision criteria. . 20

5.1. The Finite State Machine (FSM) of the CBAP Clustering Algorithm. 23
5.2. $v_{average(node)}$ calculation algorithm. 25
5.3. Performance difference between the two clustering algorithms. 26
5.4. The clustering sub-protocol HELLO packet structure. 28
5.5. CH Decision Criterion. 31
5.6. CH Selection Decision. 33
5.7. Hierarchical aggregation model. 37
5.8. An example of a Bloom filter and its operation [28]. 39
5.9. The TTL range as a function of $v_{avg(cluster)}$. 43

6.1. Node satisfying the velocity threshold condition and participating in cluster formation. 48
6.2. Dependence of the inter-vehicle contact time on the relative velocity and communication range. 49
6.3. Normal distribution of node velocities in the average traffic density scenario. 51
6.4. Cumulative distribution functions of the two clustering mechanisms for different traffic scenarios ($L = 4$). 53
6.5. Cumulative distribution functions of the two clustering mechanisms for different traffic scenarios ($L = 2$). 53
6.6. Cumulative distribution functions of the two clustering mechanisms for different traffic scenarios. 54

6.7. Dependence of cluster membership duration on the node acceleration (velocity variation). 56
6.8. Hierarchical aggregation case scenario. 57
6.9. Probability of false positives (f) as a function of the amount of CIDs combined within the CCID. 64
6.10. The optimal number of hash functions depending on the average amount of neighboring clusters (CIDs), for two different lengths of the CID array. . . . 65

List of Tables

1.1. Types of VANET Applications [8]. 7

5.1. Calculation results for $v_{average(node)}$. 26

6.1. Parameter Summary 52

Bibliography

[1] Stefan Dietzel, Mihail Balanici and Frank Kargl, “Towards Data-similarity-based Clustering for Inter-vehicle Communication,” *Vehicular Networking Conference*, December 2013.

[2] Rabah Meraihi, Sidi-Mohammed Senouci, Djamal-Eddine Meddour and Moez Jerbi, *Chapter 12. Vehicle-to-Vehicle Communications: Applications and Perspectives*. Wiley-ISTE, December 2007.

[3] European Telecommunications Standards Institute, "http://www.etsi.org/technologies-clusters/technologies/intelligent-transport".

[4] Fan Li and Yu Wangl, “Routing in Vehicular Ad Hoc Networks: A Survey,” *IEEE Vehicular Technology Magazine*, vol. 2, no. 1, June 2007.

[5] Georgios Karagiannis, Onur Altintas, Eylem Ekici, Geert Heijenk, Boangoat Jarupan, Kenneth Lin and Timothy Weil, “Vehicular Networking: A Survey and Tutorial on Requirements, Architectures, Challenges, Standards and Solutions,” *IEEE Communications Surveys & Tutorials*, vol. 13, no. 4, pp. 584–616, 2011.

[6] Car-to-car Communication Consortium, "http://www.car-2-car.org", [Online; accessed 10-October-2013].

[7] Jean-Pierre Hubaux, Srdjan Capkun and Jun Luo, “The Security and Privacy of Smart Vehicles,” *IEEE Security & Privacy*, vol. 2, no. 3, pp. 49–55, May/June 2004.

[8] Elmar Schoch, Frank Kargl, Michael Weber and Tim Leinmüller, “Communication patterns in VANETs,” *IEEE Communications Magazine*, vol. 46, no. 11, 2008.

[9] Stefan Dietzel, Boto Bako, Elmar Schoch and Frank Kargl, “A Fuzzy Logic Based Approach for Structure-free Aggregation in Vehicular Ad-hoc Networks,” in *Proceedings of the Sixth ACM International Workshop on VehicuIAr InterNETworking*, ser. VANET ’09. New York, NY, USA: ACM, 2009, pp. 79–88. [Online]. Available: http://doi.acm.org/10.1145/1614269.1614283

[10] Stefan Dietzel, Frank Kargl, Geert Heijenk and Florian Schaub, “Modeling In-Network Aggregation in VANETs,” *IEEE Communications Magazine*, vol. 49, no. 11, pp. 142–148, November 2011.

[11] Björn Scheuermann, Christian Lochert, Jedrzej Rybicki and Martin Mauve, “A Fundamental Scalability Criterion for Data Aggregation in VANETs,” in *Proceedings of the 15th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom ’09. New York, NY, USA: ACM, 2009, pp. 285–296. [Online]. Available: http://doi.acm.org/10.1145/1614320.1614352

[12] W. R. Heinzelman, A. Chandrakasan and H. Balakrishnan, “Energy-Efficient Communication Protocol for Wireless Microsensor Networks,” *HICSS’00: Proceedings of the 33rd Hawaii International Conference on System Sciences*, vol. 8, January 2000.

[13] Y. Yao and J. Gehrke, “The cougar approach to in-network query processing in sensor networks,” *SIGMOD Rec.*, vol. 31, no. 3, pp. 9–18, 2002.

[14] European Telecommunications Standards Institute, “ETSI TR 102 863 Technical Report,” Tech. Rep., June 2011.

[15] ——, “ETSI TS 102 637-2 Technical Specification,” Tech. Rep., March 2011.

[16] R. A. Santos, R.M. Edwards and N.L. Seed, “Using the Cluster-Based Location Routing (CBLR) Algorithm for Exchanging Information on a Motorway,” *4th International Workshop on Mobile and Wireless Communications Network*, pp. 212–216, Septermber 2002.

[17] Mingliang Jiang, Jinyang Li and Yong Chiang Tay, “Cluster Based Routing Protocol (CBRP) Functional Specification,” *IETF Internet Draft*, August 1998.

[18] Tao Song, Weiwei Xia, Tiecheng Song and Lianfeng Shen, “A Cluster-Based Directional Routing Protocol in VANET,” *Proceedings of the 12th IEEE International Conference on Communication Technology (ICCT)*, pp. 1172–1175, November 2010.

[19] Yuyi Luo, Wei Zhang and Yangqing Hu, “A New Cluster Based Routing Protocol for VANET,” *Second International Conference on Networks Security, Wireless Communications and Trusted Computing (NSWCTC)*, vol. 1, pp. 176–180, April 2010.

[20] Yvonne Günter, Bernhard Wiegel and Hans Peter Großmann, “Cluster-based Medium Access Scheme for VANETs,” *Proceedings of the 10th IEEE International Conference on Intelligent Transportation Systems*, pp. 343–348, September–October 2007.

[21] Evandro Souza, Ioanis Nikolaidis and Pawel Gburzynski, “A New Aggregate Local Mobility (ALM) Clustering Algorithm for VANETs,” *Proceedings of the IEEE ICC 2010*, pp. 1–5, May 2010.

[22] Zaydoun Y. Rawshdeh and Syed Masud Mahmud, “Toward Strongley Connected Clustering Structure in Vehicular Ad hoc Networks,” *IEEE Vehicular Technology Conference. Proceedings*, vol. 1, 2009.

[23] Lars Wischhof, Andre Ebner and Hermann Rohling, “Information Dissemination in Self-Organizing Intervehicle Nertworks,” *Intelligent Transportation Systems, IEEE Transactions*, vol. 6, no. 1, March 2005.

[24] Tamer Nadeem, Sasan Dashtinezhad, Chunyuan Liao and Liviu Iftode, “TrafficView: Traffic Data Dissemination using Car-to-Car Communication,” *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 8, no. 3, pp. 6–19, July 2004.

[25] Khaled Ibrahim and Michele C. Weigle, “CASCADE: Cluster-based Accurate Syntactic Compression of Aggregated Data in VANETs,” *Technical Report (available at http://www.cs.odu)*, 2008.

[26] Christian Lochert, Björn Scheuermann and Martin Mauve, “Probabilistic Aggregation for Data Dissemination in VANETs,” *VANET 2007: Proceedings of the Fourth ACM International Workshop on Vehicular Ad Hoc Networks*, pp. 1–8, September 2007.

[27] P. Flajolet and G. N. Martin, “Probabilistic counting algorithms for data base applications,” *Journal of Computer and System Sciences*, vol. 31, no. 2, pp. 182–209, October 1985.

[28] Andrei Broder and Michael Mitzenmacher, “Network Applications of Bloom Filters: A Survey,” *Internet Mathematics*, vol. 1, no. 4, pp. 485–509, 2004.

[29] Burton H. Bloom, “Space/Time Trade-offs in Hash Coding with Allowable Errors,” *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, July 1970.

[30] http://en.wikipedia.org/wiki/Bloom_filter.

[31] S. Joshua Swamidass and Pierre Baldi, “Mathematical Correction for Fingerprint Similarity Measures to Improve Chemical Retrieval,” *Journal of chemical information and modeling (ACS Publications)*, vol. 47, no. 3, pp. 952–964, 2007.

[32] Annette Böhm, Magnus Jonsson and Elisabeth Uhlemann, “Adaptive Cooperative Awareness Messaging for Enhanced Overtaking Assistance on Rural Roads,” *Proceedings of the IEEE Vehicular Technology Conference*, pp. 1–5, September 2011.

[33] European Telecommunications Standards Institute, “Draft ETSI EN 302 637-2: Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Specification of Cooperative Awareness Basic Service,” Tech. Rep., August 2013.

[34] ——, “ETSI TS 103 097: Intelligent Transport Systems (ITS); Security; Security header and certificate formats,” Tech. Rep., April 2013.

[35] S. Poisson, “Probabilité jugements en matière criminelle et en matière civile précédées règles générales calcul probabilitiés (Paris),” *Bachelier, Paris*, vol. 206, 1837.

[36] Peng Fan, Prasad Sistla and Peter C. Nelson, “Theoretical Analysis of a Directional Stability-based Clustering Algorithm for VANETs,” *Proceedings of the Fifth ACM International Workshop on VehiculAr Inter-NETworking*, pp. 80–81, September 2008.

Acronyms

AHSRA	Advanced Cruise Assist Highway Systems Research Association
AP	Access Point
ASV	Advanced Safety Vehicle
C2C-CC	Car2Car Communication Consortium
California PATH	California Partners for Advanced Transit and Highways
CAM	Cooperative Awareness Message
CBAP	Cluster-Based Aggregation Protocol
CBDRP	Cluster-Based Directional Routing Protocol
CBLR	Cluster-Based Location Routing
CBMAC	Cluster-Based Medium Access Control
CBRP	Cluster-Based Routing Protocol
CCID	Combined Cluster ID
CICAS	Cooperative Intersection Collision Avoidance System
CID	Cluster ID
CH	Cluster Head
CHP	Cluster Head Pending
COOPERS	Cooperative Systems for Intelligent Road Safety
CM	Cluster Member
CMGW	Cluster Member Gateway
CU	Cluster Undecided
CVIS	Cooperative Vehicle-Infrastructure Systems
DDA	Duplicate Detection Algorithm

ETC	Electronic Toll Collection
ETSI	European Telecommunications Standards Institute
ICT	Information and Communications Technology
IntelliDrive(sm)/VII	Intelligent Drive/Vehicle Infrastructure Integration
ITS	Intelligent Transport System
IVC	Inter-vehicle Communication
LDM	Local Dynamic Map
LTE	Long Term Evolution
MANET	Mobile Ad Hoc Network
NoW	Network on Wheels
PRECIOSA	Privacy enabled capability in co-operative systems and safety applications
RSU	Road Side Unit
SafeTrip21	Safe and Efficient Travel through Innovation and Partnership for the 21st century
SEVECOM	Secure Vehicular Communication
SOTIS	Self-organizing Traffic Information System
UMTS	Universal Mobile Telecommunications System
V2I	Vehicle-to-Infrastructure
V2V	Vehicle-to-Vehicle
V2X	Vehicle-to-X (Vehicle/Infrastructure)
VANET	Vehicular Ad Hoc Network
VICS	Vehicle Information and Communication System
VSC	Vehicle Safety Communications
WLAN	Wireless Local Area Network
WSN	Wireless Sensor Network

Name: Mihail Balanici Matriculation Number: 748378

Erklärung

Ich erkläre, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Mihail Balanici