UNIVERSITÀ DEGLI STUDI DI PALERMO

Dottorato di Ricerca in INGEGNERIA DELL'INNOVAZIONE TECNOLOGICA
Dipartimento di Ingegneria
SSD: ING-INF/05

A framework for Population Protocols in VANETs

IL DOTTORE
Ing. Antonio Bordonaro

IL COORDINATORE
Ch.mo Prof. Salvatore Gaglio

IL TUTOR
Ch.mo Prof. Giuseppe Lo Re

IL CO-TUTOR
Ch.ma Prof.ssa Alessandra De Paola

CICLO XXXV
ANNO CONSEGUIMENTO TITOLO: 2023

Contents

Abstract

Vehicular Ad Hoc Networks (VANETs) are wireless communication networks that connect moving vehicles and the supporting road infrastructure. This emerging technology can transform the way cars and road infrastructure interact, improving safety, efficiency, and quality of transportation. Despite the great potential offered by VANETs, several challenges remain unresolved. Among these, one of the most significant concerns the algorithms used for communication between nodes. Existing routing algorithms often struggle to adapt to the dynamic nature of VANETs, resulting in inefficiency nodes communication.

In this work, we analyze the possibility of adopting a different communication model from conventional routing algorithms: the Population Protocol model represents a viable alternative to the use of classical communication models. This model represents an elegant formalism for dealing with the typical problems of distributed systems. Although by its characteristics it completely satisfies the requirements of vehicular networks, its use in a real-world context required solving some problems related to the physical constraints that are considered in the theoretical model, which we solved by defining an appropriate 3-way communication scheme.

The full power of PP can be exploited in the Event Detection and Dissemination system proposed in this thesis. The proposed system is executed in the vehicular network in a fully distributed fashion: it requires no network infrastructure and no initialization phase. It adopts an Event Driven approach, in which components react asynchronously to events that occur. Experimental results show the validity of the system and how efficiently nodes are able to disseminate information within the network.

Chapter 1

Introduction

In recent years, we have witnessed a real revolution in transportation systems, which are now able to provide services that facilitate the driver's experience, or provide higher level of road safety. Among the most popular applications, we can easily find the *lane assist* technology to avoid off-tracks, or the *autonomous emergency braking* that causes the vehicle to slow down when approaching a hazard. All of these services have been made possible by means of the sensors embedded into today's vehicles. Many automakers, such as Renault, Volvo, and Volkswagen, are investigating the possibility of leveraging these sensors not only to improve consumer's experience of their products, but also to enable the deployment of a wide variety of complex services in a broader domain, such as *smart environments* [35]. For example, vehicles in the proximity of an accident could capture and propagate such information to alert about the presence of danger to all drivers traveling in that direction. Another possible example is a set of vehicles that detect the presence of vehicular traffic, and broadcast this event to other entities in the network to recommend better travel routes.

To accomplish these tasks, it is mandatory to leverage a network of interconnected vehicles that collect and share information along their routes. This idea resulted in the Vehicular Ad-hoc NETwork (VANET), a paradigm for which vehicles are modeled as nodes in a highly variable and dynamic network in which they communicate through short-range wireless technologies. In this context, commu-

nications depend on the underlying infrastructure that VANETs can rely on, and can be categorized as Vehicle-to-Infrastructure (V2I) or Vehicle-to-Vehicle (V2V).

VANETs are a special case of the more generic Mobile Ad-Hoc Networks (MANETs), defined precisely as an autonomous system of mobile terminals connected via wireless ad-hoc links. They are built as needed and used in dynamic environments, not necessarily with the support of an existing infrastructure. All nodes in the system cooperate with the purpose of routing packets in the correct way according to the multihop forwarding mode. The various attempts to adapt MANET algorithms to vehicular networks have never been successful: due to the unique characteristics of VANETs (very high node speed, sudden changes in network topology, etc.) it is not possible to adopt solutions proposed specifically for MANETs. Therefore, specific solutions for VANETs must be proposed.

The V2I VANET type leverages an infrastructure composed of several fixed units, named Road Side Units (RSUs), responsible for providing information from/to the vehicles. Although the adoption of this infrastructure result in several benefits, especially in the cybersecurity domain, V2I communications are poorly suited to real-world scenarios because of the high installation and maintenance costs. Moved by these limitations, V2V communications have been investigated. Here, the infrastructure is completely distributed as it is only composed of vehicles that interact each other according specific conditions, e.g., the physical distance between two vehicles.

The adoption of the V2V VANET technology is not trivial since several challenges have to be addressed. For example, the V2V-based networks could be subject to the packet congestion when the density of vehicles is high; conversely, the dissemination of the sensed information may take a long time if there are few vehicles in the area of interest. Moreover, V2V-based networks are prone to malicious attacks due to the lack of robust authentication mechanisms [20]. In spite of such challenges, the potential impact of VANET applications in smart environments justifies the importance of this paradigm and the significant interest in the research community. In particular, the researchers have identified that the fundamental problem of VANETs lies in the *Data Dissemination* and, consequently, how fast-moving vehicles communicate with each other [17, 53].

There are no standard approaches to alleviate these issues and several techniques have been proposed, each of which is strongly dependent on the adopted routing strategies. Some techniques exploit additional data about the vehicles' mobility, such as driving directions, in order to create clusters through which information can be efficiently forwarded; others leverage probabilistic models that determine whether or not a certain vehicle should forward the information. In general, the existing strategies add a significant computational overhead to the vehicular network, thus causing link failures or degradation in network scalability in the worst case [52].

The Population Protocols (PPs) could be a viable alternative because they do not require additional processing steps compared to state-of-the-art techniques. The Population Protocols are usually adopted in distributed scenarios and consist of a population of nodes that cooperate each other to converge to the same high-level information about an event of interest.

Having this idea in mind, this thesis work discusses, on the one hand, the definition of appropriate communication schemes that enable the use of Population Protocols in VANETs and, on the other hand, the definition of a fully distributed Event Detection and Dissemination systems that, by leveraging the power of Population Protocols, achieves optimal performance. The proposed Event Detection system is designed layer-wise so that it can have numerous advantages: high level of decoupling between layers, modularity, independence between layers, and ease of updating or modifying the various components. One of the key components of the architecture, which required extensive design analysis and on which a specific experimental evaluation was conducted, is the module that makes the entire system adaptive to the rapid changes in network density: in fact, thanks to a sophisticated adaptive message transmission algorithm, the overall system is able to maintain a low error rate regardless of the number of nodes in the network.

1.1 Contributions

This thesis work describes a novel communication schema for VANET applications based on the Population Protocol model. In essence, the contributions of this paper are as follows:

- We propose a communication scheme that enables the adoption of the Population Protocols model in a VANET scenario ensuring that the theoretical assumptions, required by the model to achieve convergence, are fulfilled;
- The proposed communication scheme ensures that vehicles converge to the same information, and allows vehicles to assume different roles during the communication, thus enabling the implementation of asymmetric algorithms;
- The proposed scheme can support VANET-based applications in which vehicles have to be efficiently updated about specific events, e.g. vehicular traffic along the road;
- To the best of our knowledge, this work is the first to apply the Population Protocol model to VANET applications. The analysis conducted in this thesis suggests that the used approach is suitable for real application scenarios, as it shows the best trade-off in terms of messages exchanged and convergence time;
- Design of a Data Dissemination system that exploits only V2V communications but requires no overhead for initialization of the communication algorithm;
- Defining an adaptive message delivery algorithm that can ensure a high rate of information exchange between nodes while maintaining a low overall error rate;
- A layer-based Event Dissemination Framework in VANETs was realized. The framework leverages only V2V communications and it requires no network infrastructure, leverages the full power of Population Protocols for communications, it also dynamically modifies its behavior to adapt to rapid changes in vehicular density.

1.2 Dissertation Outline

The thesis work is structured as follows.

Chapter 2 describes the Population Protocol model in detail, first providing a high-level description, and then presenting the theoretical concepts on which this model is based, with particular focus on constraints and theoretical requirements.

Chapter 3 presents related work and provides insights into current Event Detection approaches and describes the state of the art related to the Population Protocol model, also presenting the main application fields of use.

Chapter 4 presents the proposed Event Detection system, describing the various architecture layers. Particular focus is given to the layer that implements the adaptive algorithm for sending messages. In addition, the communication scheme that enables the use of PPs in VANETs, is presented, describing all the mechanisms of the protocol.

Chapter 5 presents the experimental evaluation conducted on both the Event Detection system and VPP. The setup, the experiments conducted and the experimental results obtained are described.

Finally, Chapter 6 presents the conclusions of the thesis and provides insights into possible future extensions.

1.3 Publications

Parts of this doctoral dissertation have been published in international conferences. Specifically:

- **Modeling efficient and effective communications in vanet through population protocols.** Bordonaro, A., Concone, F., De Paola, A., Lo Re, G., & Das, S. K. (2021, August). In 2021 IEEE International Conference on Smart Computing (SMARTCOMP) IEEE.
- **VPP: A Communication Schema for Population Protocols in VANET.** Bordonaro, A., De Paola, A., & Lo Re, G. (2021, December). In 2021 20th International Conference on Ubiquitous Computing and Communications (IUCC/CIT/DSCI/SmartCNS) IEEE.
- **Smart Auctions for Autonomic Ambient Intelligence Systems.** Bordonaro, A., De Paola, A., Lo Re, G., & Morana, M. (2020, September). In

2020 IEEE International Conference on Smart Computing (SMARTCOMP) IEEE.

- **On-board energy consumption assessment for symbolic execution models on embedded devices.** Bordonaro, A., Gaglio, S., Lo Re, G., Martorella, G., & Peri, D. (2020, September). In 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA) IEEE.

- **Adaptive Event Detection in Vehicular Networks through Population Protocols.** A. Bordonaro, F. Concone, A. De Paola, G. Lo Re, A. Virga, submitted to IEEE Access.

Chapter 2

Population Protocol Model

This chapter opens with an overview about Population Protocols, providing a solid starting point to fully comprehend the proposed approach. Then, the requirements to adapt this model to VANET applications are presented by discussing a simple scenario.

The Population Protocols (PPs) [7] were designed as a theoretical model to describe a population of resource-constrained hardware devices, namely *agents*, characterized by random movements. The goal of any PP-based algorithms is to accomplish a task by relying on random interactions between agents. For example, an important class of population protocols is focused on consensus algorithms, in which agents need to collectively reach a decision about a given information [6].

Formally, a population is composed of N agents, with $N \geq 2$, each of which is defined as an automata with a finite number of states in a state space S. Each agent initially has an input value σ from an alphabet Σ that is used by an *input mapping function* $\lambda(\cdot)$ to set its initial state $s_i \in S$. Pairwise interactions update the states of both agents according to a *transition function* $\delta(\cdot)$ that takes both states as input and returns new states for both agents as output. Here, the agents' interactions are considered (i) unpredictable because there is no knowledge about the order in which the interactions occur, and (ii) asymmetric, i.e. one of the agents is the initiator of the interaction, and one the responder. Note that for an algorithm

with an asymmetric transition function to be executed, it is necessary for the two interacting nodes to be able to assume two different roles in the communication.

The PP-based algorithms may also converge after T iterations, but agents are unable to determine if such a convergence is reached. However, each agent is able to produce an output value that describes its own perception about the surrounding environment. This information is generated by using an appropriate *output mapping function* $\Omega(\cdot)$, which maps the current state $s_i \in S$ into a value $z \in Z$, where Z is the output alphabet. These parameters are summarized in Table 2.1

At each time t, the model describes the population by means of a *configuration* C_t that contains the states of all the agents. This means that an execution of a PP-based algorithm is an infinite sequence of configurations $C_0, C_1, C_2, \ldots, C_\infty$, where $C_0 = \{\lambda(\sigma_i)\}_{i=1,n}$ is the initial configuration, and a general transition $C_t \rightarrow C_{t+1}$ occurs only when C_{t+1} can be obtained from C_t by a single interaction of two agents. The protocol is *deterministic* if there is only one possible interaction $(s_,s_2) \rightarrow (s'_1, s'_2)$ for each pair of s_1, s_2. In other words, the algorithm execution can be schematized by a sequence of configurations C_T. [10].

All the agents of a population are characterized by identical features and consequently they are indistinguishable; the system behavior is thus defined by the finite state automata which describes the behavior of the single agent.

Typically, algorithms based on the Population Protocol model are characterized by the following features [10]:

- **Agent finite-state machine**: each agent can store a finite number of bits, which does not depend on population size.
- **Uniformity**: the algorithm does not depend on the population size.
- **Computation based on interactions**: agents cannot send messages or share memory with other nodes. The only way to update agent states is through interactions. The concept of interaction is not specified in the population protocol definition but must be defined according to the specific application field. It is important to note that the interaction involves two nodes that must both update their states consistently. Fulfilling this theo-

Table 2.1: Population Protocols Model parameters

Parameter	Description
Σ	A finite sequence of symbols that can be provided as input to the agents during the algorithm initialization phase
S	A finite set of agent states
$\lambda(\sigma)$	An input mapping function which maps each $\sigma \in \Sigma$ element into an $s \in S$ element. It allows each agent that receives the input value σ, to determine its initial state s_0
Z	A finite sequence of symbols that an agent can produce as output value
$\Omega(s)$	An output mapping function which maps each element $s \in S$ into an element $z \in Z$. It allows each agent to determine its output value z, based on its current state s
$\delta(s_1, s_2) \subseteq S^4$	A transition function that accepts as input the states of two interacting agents and returns the new pair of states. Specifically, if two agents with (s_1, s_2) states interact, after the interaction they will reach the (s'_1, s'_2) pair of states

retical property, in a real context, subject to constraints of different nature, may not be possible.

- **Unpredictability of interactions**: The order in which agents interact is random. However, for the convergence of the algorithm, it is important that the interactions respect the *equity* constraint.
- **Distributed Input and Output**: Input and output values are provided and produced in a distributed way.
- **Convergence rather than termination**: agents are generally unable to determine when the algorithm achieves the convergence.

In many application fields, the inability to distinguish two nodes implies serious restrictions. Several extensions have been proposed to overcome such limitation. One of the most relevant extension in this direction is represented by the Community Protocols [37] which provide the possibility to uniquely identify agents

Table 2.2: A possible computation for the example discussed in Chapter 2

x	0	0	1	1	1	1	*input*
C_0	s_0	s_0	s_1	s_1	s_1	s_1	$\delta(s_1, s_1) = (s_0, s_2)$
C_1	s_0	s_0	s_0	s_1	s_1	s_2	$\delta(s_1, s_1) = (s_0, s_2)$
C_2	s_0	s_0	s_0	s_0	s_2	s_2	$\delta(s_2, s_2) = (s_0, s_4)$
C_3	s_0	s_0	s_0	s_0	s_0	s_4	$\delta(s_0, s_4) = (s_0, s_4)$
C_4	s_0	s_0	s_0	s_0	s_0	s_4	$\delta(s_0, s_0) = (s_0, s_0)$
...	s_0	s_0	s_0	s_0	s_0	s_4	...
y	0	0	0	0	0	0	*output*

.

through the use of identifiers. Specifically, the Community Protocols extend the Population Protocol model by introducing the following element:

- U: a set of symbols (plus a special symbol ϕ which denotes the null element) used as agents identifiers.

Population Protocols therefore represent a powerful formalism for modeling many of the typical scenarios of distributed systems. Therefore, they represent an excellent paradigm, with very high potential, to be used in the fast-growing context of VANETs

Example: Consider a simple scenario in which 6 vehicles are equipped with a sensor that returns whether their speed is high or not, and we are interested in knowing whether at least 5 vehicles have high speeds. Also assume that the sensors, at each instant t, return 1 if the speed is above a certain threshold, 0 otherwise. This problem can be addressed by a PP-based algorithm in which $\Sigma = Z = \{0, 1\}$, the input function $\lambda(s_0) = 0$ and $\lambda(s_1) = 1$, the output function is

$$\Omega(s_i) = \begin{cases} 0 & \text{if } \{s_i\}_{i=0,4} \\ 1 & \text{otherwise,} \end{cases} \tag{2.1}$$

and, finally, the transition function is

$$\delta(s_i, s_j) = \begin{cases} (s_5, s_5) & \text{if } i + j \geq 5 \\ (s_0, s_{i+j}) & \text{otherwise.} \end{cases} \tag{2.2}$$

Table 2.2 shows a possible computation for the algorithm. At instant t, the sensors return the input $x = \{0, 0, 1, 1, 1, 1\}$ that is mapped, by the input function, into the configuration C_0. A possible next configuration may be C_1, i.e. the one generated by the interaction between the 3-rd and 6-th vehicles. This computation can run indefinitely, but Table 2.2 highlights that after the configuration C_3 is achieved, the outcome will be always the same. In other words, $T = 4$ iterations are sufficient for the algorithm convergence allowing for the output mapping function to be applied and terminate the computation. Regardless this aspect, after the fourth iteration the perception of each vehicle is that there are not at least 5 vehicles with high speed, i.e. $y = \{0, 0, 0, 0, 0, 0\}$.

Chapter 3

Related Work

VANETs [58] represent a particular case of Mobile Ad-hoc Networks (MANETs) and play a key role in Intelligent Transportation Systems (ITS) currently representing an attractive opportunity to improve road safety and passenger comfort [39]. Several challenges are still open, related to security [46], infrastructure and communication models [34]. Due to the intrinsic nature of the vehicular networking, typically, the development of VANET applications requires nodes to exchange information. Therefore, one of the main challenges concerns the efficient communication between nodes [24]. Several approaches have been investigated over the last year, each of which show advantages and disadvantages [38]. In theory, algorithms designed for MANETs could be used. However, as it is well known, VANET nodes' high mobility motivates research for the design of ad-hoc algorithms and communication models. Current VANET algorithms are essentially based on the following different communication models: *Road Side Unit Communication Model*, *Vehicle to Vehicle Communication Model*, and *Cluster Based Communication Model* [38].

According to the *Road Side Unit Communication Model*, nodes communicate through Road Side Units (RSUs), trusted entities installed in zones of interest, which operate as routers in the network. Although this approach has several advantages (the partially centralized architecture that simplifies node management and coordination, or some extra security guarantees, since messages between nodes flow only through the RSU), it is affected by several limitations. First of all, the

necessity of an infrastructure (consisting of RSUs) covering the whole VANET operating area. Namely, due to physical constraints and high deployment costs, in many cases it is not possible to install the RSUs pervasively [5]. Furthermore, RSUs represent, with respect to the controlled nodes, a single point of failure. This means that if a RSU fails or cannot manage all the received requests, nodes connected through will experience disconnections.

In the *Vehicle to Vehicle Communication Model*, each vehicle is equipped with an *On Board Unit* (OBU), which allows communication with other vehicles. This model overcomes many of the limitations of the RSU-based model (e.g., it does not require ad hoc infrastructure, significantly lowering costs), however, it introduces others. One of the main concern is the Multi-hop communication between nodes. Namely, the communication between a source and a destination is made possible by other intermediary nodes [42].

In the *Cluster Based Communication Model* vehicles are grouped into different *clusters*. Within each cluster, a client-server architecture is established, with a head node acting as a server and the other nodes as clients. Communications between clusters occur only through the head nodes, which, subsequently, send information to their clients. This architecture overcomes the limitations of the multi-hop model and has several advantages, such as the possibility to adopt a communication schema independent from any infrastructure and a reduced network failure rate, however, it introduces new limitations. The main limitations concern the overhead required for the head election and the strong dependency of the intra-cluster communication on the connectivity with the cluster head.

Although the Vehicle to Vehicle Communication Model is the most promising for the development of current VANET applications, it has however some limitations that weaken its application in different scenarios. Among these, a relevant limitation is that the performance are strongly affected by the routing protocols adopted to deliver messages from a source to a destination. Typically, these protocols are developed specifically for the application in which they are used, but since the VANET scenario is highly dynamic, they are generally unable to quickly adapt to sudden changes in network (topology, density, speed) [31]. For this reason, there is a need for an efficient model that can, on the one hand, overcome the limita-

tions of RSU-based models and, on the other hand, fully exploit the potential of a Vehicle to Vehicle Communication Model.

Many works have been proposed with the aim of designing innovative systems able detect events in VANETs. In [60], the authors propose a novel mechanism to get insight into VANET messages to detect different levels of traffic jams. In particular, beacon messages generated by each vehicle are modeled as a position and speed change event, which is processed by an event processing agent to detect different levels of congestion on a road. A more recent and interesting work is discussed in [4]. Here, a framework for the real-time distributed classification of non-recurrent congestions is presented. Each congestion event is modeled by unique features extracted by means of the vehicle's context, and then used for the final classification using a machine learning algorithm.

All of these works achieve high performance in recognizing an event of interest, but nothing is mentioned about how the event is disseminated. In fact, this phase should not be neglected as it can negatively affect the accuracy of detection, and other aspects [49]. As an example, consider the scenario addressed in [9]. Assume that a vehicle has detected an event in the proximity, and it begins to share that event with the rest of the network at a predetermined frequency. Then, each neighboring vehicle receives the message and forwards it to its neighbors, and so on. It is clear that such a dissemination strategy inevitably leads to several problems [16] (e.g., network congestion, message loss, signal loss) that affect the vehicle's ability to identify the event in its range of interest. For this reason, different *adaptive* strategies have been proposed to address the data dissemination problem in VANET in order to guarantee simultaneously real-time, reliability, and robustness requirements [59, 51, 1]. Unfortunately, despite their effectiveness, the more complex strategies require a modest overhead to determine how the information should be disseminated [55]. For example, if a cluster-based algorithm were to be used (so that event information is sent only from the head), it would be necessary first to run the clustering algorithm in a fully distributed manner, so that each node knows which cluster it belongs to.

That said, these approaches can be leveraged for the implementation of innovative event detection systems. Previously, if one wanted to detect areas of high vehicle density, or where an accident had occurred, it was necessary to rely on an

infrastructure of sensors located at different points in the urban (or extra-urban) area. Approaches based on this idea [18, 62] are quite limited because of the high cost of management and, more importantly, because their applicability is restricted to the areas where these sensors are placed. Today, leveraging approaches based on V2V communications, a network of vehicles has the potential to recognize any kind of event in a distributed manner, without the adoption of any external infrastructure. A vehicle can be considered as a dynamic entity that moves around the urban area, collects and processes the data locally to share with other entities over the network [21]. This phenomenon paves the way to scenarios in which a multitude of entities collaborates to solve a wide range of problems sharing real-time information among themselves, from public safety [44] to the monitoring of the road surface conditions [2].

The Population Protocols (PPs) could be a viable alternative because they do not require additional processing steps compared to state-of-the-art techniques. Moreover, this paradigm models interactions between nodes in a population to enable all of them to converge on a common state; this means that PPs intrinsically are able to manage both the data dissemination and the event detection tasks. The computational power of this model has been investigated, and it has been proved that the Population Protocols can solve the first order predicates of Presburger's arithmetic [8]. The Population Protocol model provides an effective and elegant formalism to deal with typical tasks of distributed systems. For instance, the authors of [11] presented an algorithm to solve the counting problem, while in [12] is defined a solution to the majority problem. The problem of the leader election through the Population Protocol is treated in [33, 57].

The original model suffers expressiveness limitations, solved by further extensions proposed in successive works [48]. For example, the authors of [37] introduces the *Community Protocols*, which extend Population Protocols by introducing identifiers. This allows to model scenarios in which nodes must be uniquely identified. Instead, in [47] were presented the *Mediated Population Protocols*, which extend the original model by introducing states associated with the network edge.

The Population Protocol model requires theoretical assumptions that in many real-world scenarios may not be fulfilled. In that regard, several works have been proposed to extended the model taking into account constraints related to the

physical world restrictions. For example, many working mechanisms of the models related to real-world properties are not defined (e.g., the way interactions occur) but the implementation is dependent on the specific scenario [37, 47]. In [54], the PP model is extended by introducing the possibility to modify the relative speed of nodes in order to evaluate how the probability of interaction and convergence time change. Another strong assumption of Population Protocols is the uniform distribution of the probability of interaction. This assumption is not always respected in a real network; for example, authors of [63] analyze the validity of this assumption in a data-collection application scenario. Instead, the authors of [32] introduce the possibility of failures in transmissions, evaluating how this impacts the convergence of the algorithm.

Thus, adapting the PP model to the vehicular context is not trivial. Other constraints include the speed of the vehicles [54] or transmission failures [32]. Our previous studies reveal that two main problems have to be solved when using PPs in VANETs, i.e., the consistent updating of states and the roles two vehicles take during an interaction [13]. These requirements can be satisfied by defining an appropriate 3-way communication scheme [14]. The scheme implements mechanisms that enable nodes to perform in the correct way the interactions expected by the Population Protocols theoretical model. This scheme, in addition to enabling the execution of asymmetric algorithms (i.e., with different transition function for the two interacting nodes), also enables convergence values characterized by low or zero error values compared to the correct theoretical convergence value. The appropriate number of messages is obtained through an experimental analysis comparing similar schemes but with a different number of message exchanges: it is shown that three is the right trade-off for good performance both in terms of efficiency on convergence times and on the error committed on the convergence value produced.

Therefore, thanks to the PP model, an event detection system is presented in this thesis that 1) succeeds in overcoming the performance limitations of current routing algorithms, 2) exploits only V2V communications and therefore does not require infrastructure, and 3) implements an adaptive communications mechanism that can quickly adapt to current network characteristics.

Chapter 4

Event Dissemination on VANET through Population Protocol

The proposed system aims to achieve a twofold goal: the first one consists in detecting specific events that may occur in the network, while the second one consists in propagating the detected events to all nodes that could take advantage, in some way, from the knowledge of the event itself. The final high-level goal is that, each node, learning that a certain event has occurred, can take advantage by implementing specific actions or changing its behavior.

For each node in the network to be aware of the occurrence of an event, nodes must continuously communicate and exchange information as efficiently as possible, a requirement met through the use of the *Population Protocol* model.

Furthermore, in order to fulfill specific performance requirements, the system is able to adapt external communications based on analysis performed on the current conditions of the surrounding network. The goal is for each node to be able to modulate, in the appropriate manner, the amount of information to be input into the network to ensure a high level of information exchange with the rest of the other nodes, while at the same time avoiding congestion in the network.

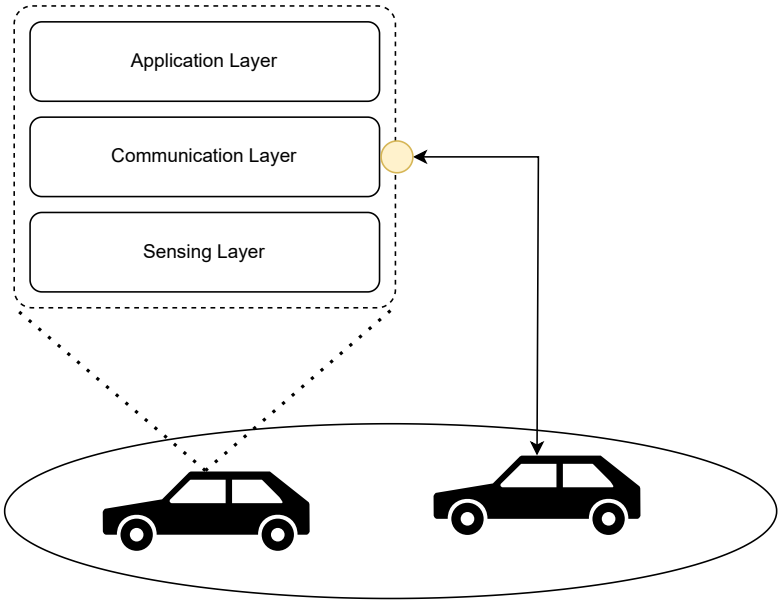


Figure 4.1: Logical architecture of the Adaptive Event Detection System.

4.1 Architecture

The system consists of a layered architecture that ensures a loose coupling between the functionality implemented by each layer.

The system adopts an *Event Driven Architecture* (EDA), which consists of producing, consuming, detecting and reacting to the occurrence of specific events. An *Event* represents a significant change in the current state of the node, or the surrounding world, which occurrence implies a specific action by the system. Examples of events might be *"An accident has occurred at 150m"* or *"The vehicle in front of us has made an unexpected stop"*. As described in the next section, the events that the system will process will be distinguished into *Low-Level Event* (LLE) and *High-Level Event* (HLE). While LLEs represent basic, simple events that cannot be broken down further, HLEs represent high-level events, produced as a function of the combination of multiple LLEs.

The architecture consists of the **Application Layer**, the **Communication Layer**, and the **Sensing Layer**, as shown in Fig. 4.1. These three layers, while cooperating and exchanging information, are independent of each other. The main

advantage is certainly the independence of the provided functionality from the actual implementation.

The *Sensing Layer* includes and models the vehicle's sensory apparatus (accelerometer, GPS, etc.) and collects sensory information from the surrounding environment. An appropriate *Data fusion* algorithm [27, 28, 25, 29, 26, 30] will process the raw sensory data to generate the *Low-Level Events*. LLEs represent simple, basic events that can be inferred from simple analyses performed locally on the sensory data. Each LLE includes, in addition to the main information it intends to model, specific *attributes*, different for each event, that define the event itself in more detail. For example, an LLE might be *Speed Decrease* which might contain the attributes *previous speed*, *current speed*, *timestamp*, etc.

The event, properly encoded, is transmitted to the *Communication Layer*, which will update the *state* of the **Population Protocol Module**.

The Communication Layer, will forward the new state to the Application Layer, and propagate it to the other nodes in the network. The communications will be modulated by the *Adaptive Communication Module*, which will adapt the sending rate in order to minimize energy consumption and the probability of network congestion, but ensure that the other nodes still receive the right amount of information. Nodes exchange messages will follow the VPP communication schema, to ensure that interactions between nodes take place in the correct way.

The *Application Layer*, having received the state from the lower layer, will give it as input to **Complex Event Processing** which, through specific methods (presumably, rule-based system), will generate the *High-Level Events* (HLEs) based on the LLEs. HLEs are a function of one or more LLEs. For example, the HLE *Traffic Jam* could be generated by a sequence of LLEs *decrease in speed*. Or, the HLE *Ostacle on roadway* event could be generated by the combination of multiple LLEs *decrease in speed* and *brusque steering*. HLEs, unlike LLEs, represent high-level information that is directly usable by *application services* and, therefore, can also be communicated to the user.

The following sections describe the layers of the architecture in detail.

4.2 Application Layer

The *Application Layer*, shown in Fig.4.2, represents the top layer of the system as well as the interface through which the user interacts with it.

It is the highest level component that collects and processes the information received from the other layers trying to communicate to the user only the information that is *relevant* for the user. The concept of *relevant*, as detailed below, is realized through the definition of appropriate decision-making policies adopted by the Application Layer.

The main functions implemented by the Application Layer are listed below:

- **Service Delivery:** the system exposes different services to the user. The user can interact with the Application Layer, through appropriate interfaces, to customize the *behavior* of individual services by setting his or her own preferences for each service;

- **Integration with Smart Environments:** The Application Layer, or its individual services, can interact and integrate with existing infrastructures in the area, if any. This has a twofold purpose: on the one hand, services provided by the vehicle to the user can be extended or improved; on the other hand, the vehicle can actively contribute to improve the functionality of the Smart Environments.

A detailed description of the services is discussed follows.

Services Delivery: The Application Layer, exploiting information received from the external world (thus from other vehicles or the other layers), will provide various high-level *Services* to the user. Examples of services might be *Smart Parking* (search, locate and navigate to free parking spaces), as well as *Smart Traffic* (traffic directions based on current traffic conditions), or even *Smart Surveillance* (if the urban infrastructure allows it, the system will integrate with the video surveillance system deployed in the area).

An *Event Driven* approach is taken, as each service is delivered and depends on the occurrence of specific events.

Each service is affected by specific events (for example, the service *SmartParking*, could be affected by the event *A vehicle has left a parking zone*). Note that a single Event could affect multiple services simultaneously. For example, assuming that the systems implements the services *Traffic Management Service (TMS)* and *Accident Detection Service (ADS)*, the event *Accident at location (X,Y)* could affect both services: the TMS could suggest taking an alternative route than the one being traveled, the ADS could simply notify the user that an accident has occurred. This organization makes the services independent and completely autonomous. Therefore, the system does not require the presence of a module that *orchestrates* the services.

According to this approach, each service could be represented as a function of the Events affecting it:

$$S_i = F(\{E_j\}_{E_j \in G_i})$$

Where S_i represents the i-th service, while the generic event E_j belongs to the set G_i, i.e., the group of events affecting the service S_i. Obviously, according to the above, any single event can affect multiple services, so it may happen that:

$$E_i \in G_j, G_k \quad j \neq k$$

The Application Layer, as shown in Fig. 4.2, consists of the following components:

- **Communication Layer Interface:** Based on the type of event received from the Communication Layer, it forwards it to the *controller* of the affected services. It can be thought of as a "mask" system. Or, it can simply forward all events to all controllers: the controllers will then discard all events that do not affect the service they control.
- **Complex Event Processing:** Complex Event Processing (CEP) receives as input the LLEs from the lower layer and, through specific inference al-

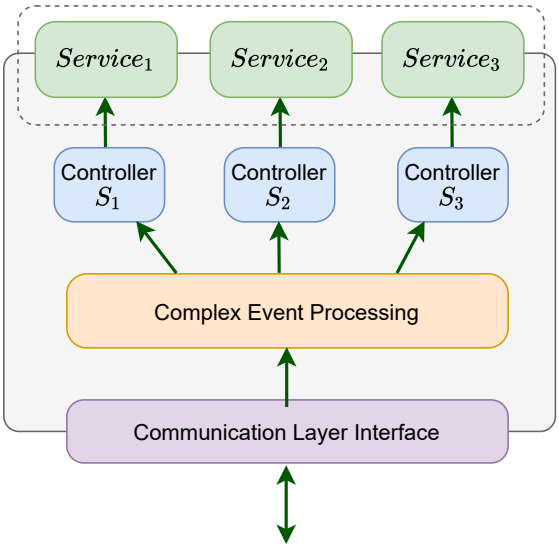


Figure 4.2: Application Layer Components

gorithms, will produce the HLEs. It will be described in detail in the next section.

- **Service Controller:** each implemented application service has an associated *Controller*. The Controller regulates, manages, organizes, and structures the events that it must forward to the relevant service, also based on the *preferences* set by the user. The main function is to determine whether a specific event, received from the lower layer, should be considered relevant and therefore sent to the pertinent service. It is in these modules that the concept of *relevance* previously anticipated, is implemented. For example, assuming that S_1 is the *Smart Parking* service, and the event *A parking space at the (X, Y)* poition is defined, the controller might decide that the location (X,Y), according to the preferences the user has set, is not relevant to the user, and therefore not forward it to the S_1 service. The controller implements appropriate **decision strategies**. Specifically, strategies that could be used are:
  - **Time-based:** if each event is also associated with the temporal sensing information, the application, through a more or less complex threshold

mechanisms, can discard events that are *temporally distant* and which, therefore, may not be useful to the user.

- **Based on location:** as in the previous case, if location information is associated with the event, the application may decide to discard events that have been detected *geographically distant* from the user's current location. In this case, the vehicle must be able to calculate its own position by means of GPS sensors.
- **Hybrid Strategies:** through specific algorithms, both temporal and location information are used to determine whether a given event may be relevant to the user. For example, if event X arrives at the node, which is geographically distant (hence location analysis) but, the vehicle estimates that, based on its current position and speed, it will be in a geographically relevant location shortly (temporal validity), then a hybrid decision policy might decide not to discard the event.

In all cases, if the event position cannot be retrieved, some mechanisms can be implemented that attempt to estimate relative positions between the node and the event. Basically, meta-information is added to the event to make sure that it is propagated in a *controlled* way, i.e., only to a part of the network (e.g., only to nodes that are within a certain *hop* distance). One technique is to associate the event with a *Time-To-Live* (TTL) value, which will be updated by the nodes as the event propagates through the network.

The controller will communicate with its *Service* through an appropriate interface. For example, functions could be defined such as `add_event(E)`, `remove_event(E)` or `update_event(E)`.

- **Service:** The *Service* processes, elaborates and maintains the information about the *events currently of interest to the user* received from the controller, communicating to the user the relevant information.

Integration with Smart Environments: The architecture is fully distributed and requires no infrastructure. Nevertheless, if the surrounding environment is equipped with smart devices (so the vehicle moves to a smart cities, or

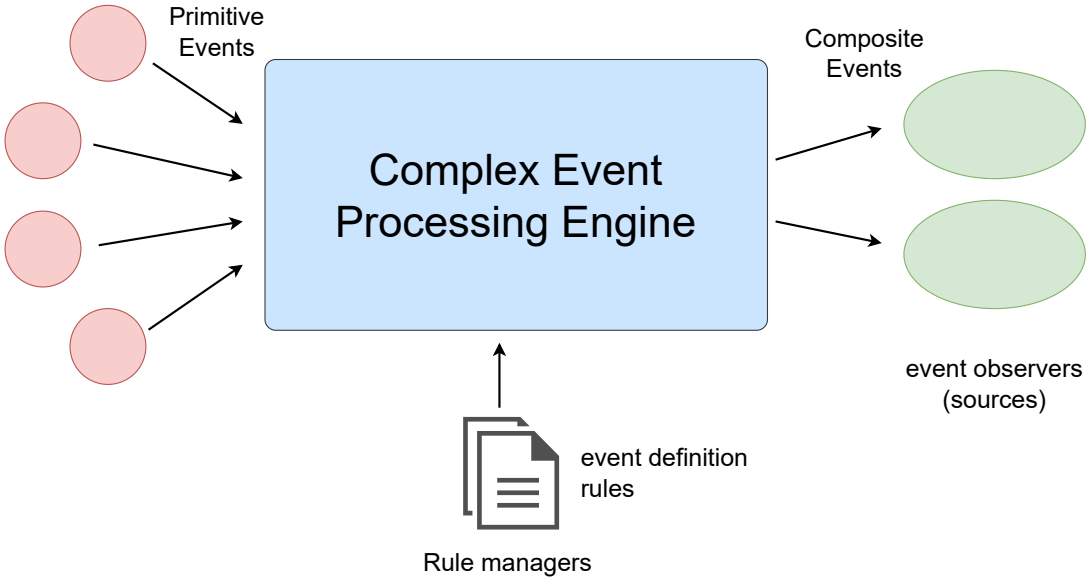


Figure 4.3: Complex Event Processing

a smart campus, or a smart parking lot), the vehicle can, through the Application Layer, interact with the surrounding environment. From this integration, both the Smart Environment and the vehicular network can benefit. For example, if the *Traffic Management System* service, delivered by the Application Layer can, in some way, take advantage of the *Smart Surveillance System* service delivered by the Smart Environment, then an interaction aimed at the exchange of information between the two services can be started.

4.2.1 CEP - Complex Event Processing

The **Complex Event Processing** (CEP) has the function of receiving *primitive events* (or LLEs) and generating *composite events* (or HLEs), which represent aggregated, high-level information. A representation of CEP, extracted from [23], is shown in Fig4.3.

The main components of a CEP are represented by the *Rule Managers* and the *Complex Event Processing Engine*. While the various Rule Managers (or a single one) have the function of managing and organizing rules, the Complex Event Processing Engine processes and analyzes LLEs to detect and determine the occurrence of HLEs, based on the defined rules.

Several CEP systems have been proposed in the literature, each defining its own data model, rule definition language and processing algorithm. Despite this, however, all systems can be categorized into two major macrocategories:

- **Data Stream Management Systems (DSMSs):** CEP systems based on computational models similar to relational databases, extended with specific operators to support on-the-fly computation. The rules of these systems transform input information (seen as streams) into one or more output streams;
- **Event/Pattern Based Systems:** rules specify how composite events (HLEs) are defined from specific *patterns* of primitive events (LLEs). These rules do not explicitly define how to transform the input stream, but the processing to be done is defined *implicitly* by the patterns. An example is the Tesla language.

Regardless of the type, each CEP defines the operators *selection*, to select specific LLEs based on their content; the operator *combination*, to aggregate different events based on mutual relationships and the temporal instants at which they occur; the operator *negation*, to identify events that have not occurred; and the operator *production*, for generating HLEs.

At present, the system uses TESLA [22], which has been used in several works [40].

In TESLA, each primitive event (LLE) is characterized by a specific *type*, which defines the number, name, and type of its attributes. Other metadata (such as, for example, timestamp) can also be associated with each event. For the sake of clarity of exposition, the example presented by [23] is reported. To model the fact that the temperature at time $T = 10$ in the tunnel at km 16.2 is 24.5C°, an LLE of the type could be defined:

```
Temp@10(km=16.2, value=24.5)
```

Where `Temp` represents the type of LLE, which includes two attributes (km and value). It can be assumed that events arrive time-ordered to the system. If not, it would be trivial to provide a module that preprocesses the events and provides

them to the CEP in order. In this case, in order to give more weight to more recent events, a time threshold could be defined beyond which to discard received events.

Thus, the TESLA rules that allow HLE events to be produced on the basis of the received LLEs have the following structure.

```
Rule R
define      ComplexEvent(att_1:Type_1, ..., att_n:Type_n)
from        Pattern
where       att_1 = f_1, .., att_n = f_n
consuming   e_1, .., e_n
```

The clause `define` introduces the new ComplexEvent being defined. The `where` clause, on the other hand, represents the attribute values of the new event, based on the functions of *aggregation* `f_1, ...f_n`, which depend on the arguments defined in `Pattern`. Together, the clauses `define` and `where`, implement the operator *production*, which precisely produces a new complex event. The `consuming` clause defines the primitive events (LLEs) that will be consumed and, therefore, not available for use by other rules.

To simplify the understanding of the language, one last example is given:

```
Rule R1
define
  TVS_Malfun(km:double, temp:double, ox:double)
from
  Oxygen(concentr=<18% and km=$a) and
  last Temp($a-10 < km < $a+10 and value>30)
  within 5 min. from Oxygen
where
  km=Oxygen.km    and
  temp=Temp.value and
  ox=Oxygen.concentr
```

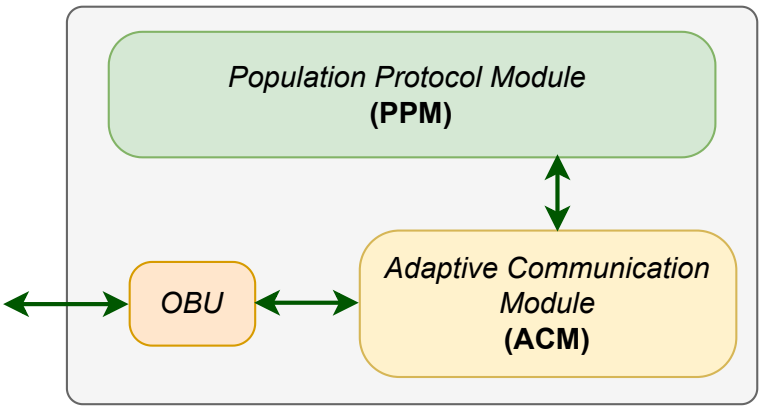


Figure 4.4: The three main components of the *Communication Layer*.

The scenario is similar to the previous example: we want to define a rule that produces the Malfunction event when the oxygen percentage, inside a tunnel, decreases below 18% and the temperature from the `a-10` to `a+10` is above 30 degrees.

The rule R1 is set, which defines the HLE event `TVS_Malfun`. This, is produced when the event `Oxygen` occurs, referred to a concentration less than 18% and at km `a` and, the temperature in the range from km `a-10` to km `a+10` is greater than 30 degrees. It can be seen that Temp and Oxygen events are related by the parameter `a`, while the operator **`last-within`** associates each Oxygen event with the last observed Temp event. In short, it relates the Oxygen events to the last Temp event.

4.3 Communication Layer

The *Communication Layer* implements all the functionalities required for the communications with other vehicles, or entities along the road. This layer is mainly composed of the three components depicted in Fig. 4.4, namely *OBU*, the *Adaptive Communication Module* (ACM), and the *Population Protocol Module* (PPM). These modules are described in detail in the following sections.

4.3.1 The Population Protocol Module

The Population Protocol Module (PPM) represents the core of the communication layer and implements the *Data Dissemination* algorithm which disseminates events among network nodes. The description of the Data Dissemination algorithm is based on the definition of all the parameters of the Population Protocol model, described below.

Input Alphabet

The input alphabet Σ is defined as follows:

$$\Sigma = \{id\}$$

where *id* is an M-bit symbol that represents the node identifier assigned during the initialization phase of the algorithm. Thus, up to 2^n different identifiers can be defined in the network. Since *id* is not used for critical functions but only for efficiency reasons, any *id collisions* doesn't cause problems.

State encoding and set of states

As described in Chapter 2, each node can be considered as an automa that, at any instant, is in a specific state. At a high level, state can be encoded as the set of information that the node must process during the execution of the algorithm.

Therefore, the state is encoded as:

$$s = \{id,\ E,\ src\}$$

Where:

- **id:** represents the vehicle identifier. It is assigned during the initialization phase of the algorithm and is kept constant during the entire execution of the algorithm. It's used by the Adaptive Communication Module to estimate the density level of the network.
- **E:** vector that contains all events that the node has collected. The event encoding is described in the following section.
- **src:** since the PPM (in the most general case) has only one hardware interface to receive states, the *src* value is used to specify the source of the received state. Specifically, in our case it is a binary value and allows to

differentiate the states coming from the vehicle itself ($sf = 1$, generated by its sensing layer) from those coming from other vehicles ($sf = 0$). This mechanism allows to adopt the architecture in different scenarios where it is necessary to distinguish multiple event sources (such as smart cities devices, RSU, etc.).

Event encoding

Events are stored in a finite-size list. Since there are no particular constraints on the memory size of vehicles, it can be assumed that an indefinite number of events can be stored, avoiding the need to implement a *replacement policy* (in the simplest case, it would be possible to remove the least recent or least relevant event).

The events vector is defined as follows

$$E = \{e_i\}_{i=1,M}$$

Where each event e_i is defined as:

$$e_i = \{Class,\, id,\, TTL,\, src,\, x,\, y,\, T\}\,;$$

Where:

- **Class:** specifies the event type. It is possible to define a specific event ontology that depends on the specific scenario and on the type of events that the system must be able to process or adopt one proposed in the literature, such as the one in [3].
- **id**: id of the node that detected the event.
- **TTL**: represents the *time to live* of the event. It is used by the Application Module to estimate the location of the event when GPS information is not available.
- **src**: similarly to the state attribute src, it is used to discriminate different source events. Specifically, it's a binary value that discriminates the events detected by the node itself ($src = 1$) from those coming from other nodes ($src = 0$).

- **x, y**: spatial coordinates. They are used only when GPS information is available.
- **T:** timestamp indicating the time instant at which the event occurred (or was detected). This could be the 32-bit Unix Epoch Time.

When GPS information is available, a node can determine whether a specific event e is relevant comparing its own location and the x, y attributes of the event. Instead, when GPS information is not available, the node estimates its relative distance to an event through a mechanism that exploits the TTL attribute. Specifically, when a node detects the event, it initializes TTL=TTL_max. The TTL_max value might depend on the type of event: some events might be associated with a larger TTL_max value, in order to spread it over a wider area. In our case, $TTL\_max = 3$. Whenever the event is received by a node, the node will append it to its event vector by decrementing the value of TTL by 1. For example, assume that s_1 and s_2 are the states of two interacting nodes. If s_1 contains event e_1 with $e_1.TTL = 3$, s_2 will append event e_1 with $TTL = 2$. Through this mechanism, the event will only propagate within a certain distance from the detection point.

Input Mapping and Output Mapping function

The input mapping function defines the initial state of a node based on its input value. Specifically, if $\sigma_i \in \Sigma$ is the input value provided to node a_i, the initial state of node a_i will be equal to $l(\sigma_i)$. In our case, the input mapping function will assign the node an *identifier* among those available in the input alphabet. For these reasons, the input mapping function, is defined as follows:

$$l(\sigma_i) = \{\sigma_i,\, E = \emptyset,\, src = 1\}; \tag{4.1}$$

Where σ_i is the identifier, E is the event vector and src is the flag indicating that it is an internal state.

The output function returns the value produced by the node based on its current state. In our case, the output function extracts the event vector from

the state, that will be consumed by the Application Layer which will report any relevant events to the user. Thus, the output mapping function is defined as:

$$w(s_i) = w(\{id,\ E,\ src\}) = E. \tag{4.2}$$

Algorithm 1: Pseudocode of the Transition Function

```
1  Function TransitionFunction(rx_state, tx_state)
2      foreach e_i in tx_state → E do
3          if e_i.TTL ≤ 1 then
4              discard e_i
5              return;
6          /* Check if tx_state is internal or external state */
7          if e_i.src == 1 then
8              /* Internal State */
9              if e_i in rx_state → E then
10                 /* Event already present in rx_state */
11                 rx_state.E.e_i.src = 0;
12                 rx_state.E.e_i.TTL = TTL_max;
13             else
14                 /* Event not present in rx_state */
15                 rx_state.E.add(e_i)
16                 rx_state.E.e_i.src = 1
17         else
18             /* External State */
19             if e_i in rx_state → E then
20                 /* Event already present in rx_state */
21                 rx_state.E.e_i.TTL = e_i.TTL
22                 rx_state.E.e_i.src = 0;
23             else
24                 /* Event not present in rx_state */
25                 rx_state.E.add(e_i)
26                 rx_state.E.e_i.src = 0
```

Transition Function

The transition function, whose pseudocode is shown in Algorithm 1, represents the core of the entire PP model, and defines the high-level logic of the Data

Dissemination algorithm. Furthermore, the transition function is the only way nodes can update their states.

In our case, the goal of the algorithm is to propagate events among the nodes in the network. Dissemination of the event should be performed as efficiently as possible. Since the exact way in which interactions take place are not defined by the theoretical model of Population Protocols, the VPP communication schema [13] can be adopted to achieve the following advantages:

- Guarantee that nodes will only update their states once they have both received the other node's state;
- Distinguish the roles of the two interacting nodes, thus an asymmetric transition function can be used. Specifically, the TX role (the node that sends the first message of the schema) and the RX role (the node that receives the first message) are defined.

If s_rx is the state of the RX node and s_tx is the state of the TX node, the δ_{RX} function (applied to the state of the RX node) operates as follows:

- if $s\_tx.src = 1$ (internal state, coming from its set of sensors): All events in s_rx are added to s_tx. If an event is already present in s_rx, it is updated by setting $e.TTL = TTL\_max$. All events are added with $e.src = 1$;
- if $s\_tx.src = 0$ (External state): All events of s_tx with $TTL > 1$ which are not contained in s_rx, will be added in s_rx with $e.src = 0$;

4.3.2 Inter-node communications: VPP

Event propagation occurs through the communications that nodes perform. As a result of these information exchanges, the nodes will update their state according to the transition function defined in the previous section. To ensure that the PP algorithm produces correct output results, it is necessary that the *interactions* occur in the right way. This is possible when the assumptions required by the theoretical PP model are fulfilled.

To meet these requirements, communications take place by adopting the communication scheme presented in [13]. This section describes the VPP protocol in detail.

VPP: A Population Protocol for VANET The specific considered scenario, i.e. a vehicular network populated by agents representing moving vehicles, has some differences compared to the abstract model of Population Protocols, thus it is necessary to introduce the following two considerations:

- **Consistent update of states**: in the theoretical model of population protocols, nodes are interconnected by defining interactions graph, which are assumed to be complete (all nodes are interconnected with each other), and the interactions occur randomly. Also, any interaction between nodes is assumed to be successful (both nodes update their states consistently). Unlike the theoretic model, in a real VANET, this assumption cannot be guaranteed. For instance, interferences, communication errors, physical obstacles, different communication ranges can produce two different results in interacting agents. Without the guarantee that the status update is consistent, the algorithm may not behave correctly and produce the correct output values.

- **Different roles of agents involved in the interaction**: In the theoretic model, the communication protocol allows nodes to assume different roles. Specifically, if there are two different roles, i.e., *role A* and *role B*, the transition function can be expressed as follows:

$$(a', b') = f(a, b) \rightarrow \begin{cases} a' = f_A(a, b) & \text{if } \textit{role A} \\ b' = f_B(a, b) & \text{if } \textit{role B} \end{cases} \tag{4.3}$$

  Successful implementation of this transition rule requires nodes to be able to determine their role in the communication, to take appropriate action. If both nodes assumed the same role, their state would not be updated correctly.

The first point states that an interaction between two nodes should only occur when they both have all the necessary information, i.e., the state of the other node.

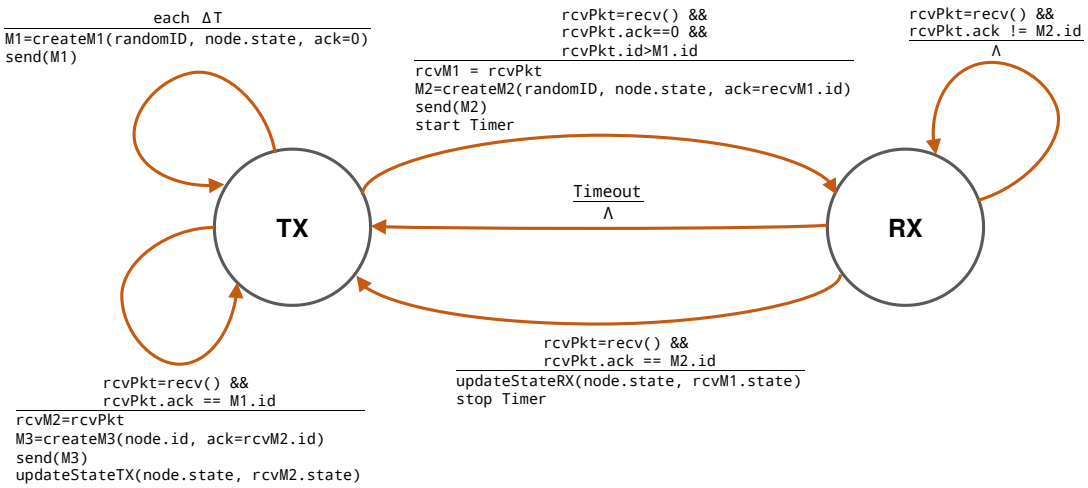


Figure 4.5: VPP Finite state machine.

This may not happen if a transmission error occurs during the state exchange or if nodes have different communication ranges. If only a node updates its state, the interaction cannot be considered correct and the algorithm will produce incorrect output values. The PP for VANET (VPP) addresses this problem through the adoption of acknowledgment messages.

Regarding the second point, VPP must support different agent roles, ensuring that two interacting nodes never play the same role. Thus, our solution supports model asymmetric communications, distinguishing between the vehicle that acts as the transmitter node (TX) and the other that acts as the receiver (RX).

The resulting communication protocol proceeds as represented by the finite state machine shown in Fig. 4.5. Each node periodically broadcasts a message (state TX in Fig. 4.5) containing its state value. During this broadcast phase, a node can receive state messages from other nodes within its communication range. To introduce the communication asymmetry necessary to perform the state update described above, each node selects a random identifier for each first message, and it plays the RX role only if it receives a state message from a node with a higher identifier (see the transition from state TX to state RX in Fig. 4.5). In this case, the RX node replies with an acknowledge message to the TX node. A node plays the TX role when it receives the acknowledge message from another node. At this point, the TX role knows the other node's state and can perform the update state

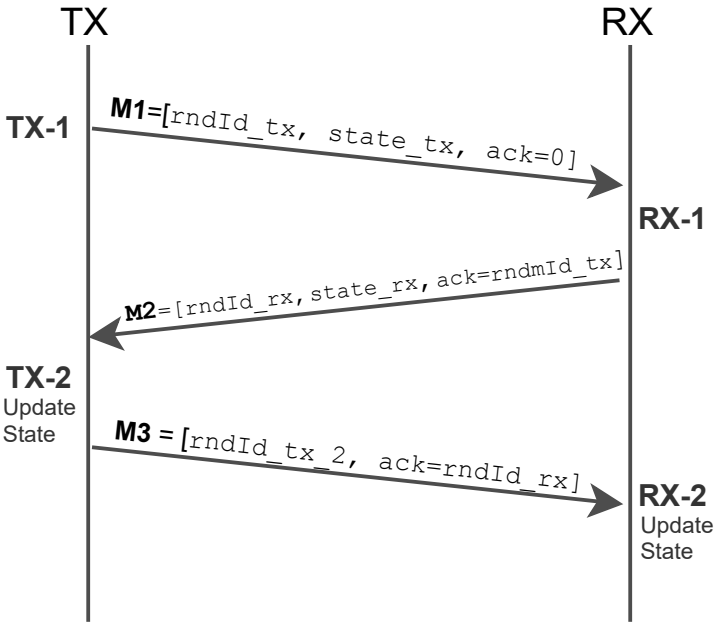


Figure 4.6: VPP Sequence diagram.

function corresponding to its role. As a final step, the TX node sends a further acknowledge message to the RX node to communicate the correct reception of its state. The receipt of this message triggers the execution of the state update function by the RX node. The resulting sequence of messages is summarized in Fig. 4.6.

Since nodes exchange messages through an unreliable channel, it is possible that some of the messages do not reach their destination. If the M1 message is not transmitted correctly, the protocol does not start and, therefore, no node updates its state. In this case, the system remains in a consistent state. If the M2 message is lost, the two nodes will stop the protocol after a certain time, consequently no nodes update their state and the system remains in a consistent state. If, on the other hand, the M3 message is lost, the TX node updates its state, but the RX node does not perform its state update, thus bringing the system to an inconsistent state.

It is worth noticing that the well-known impossibility to define a communication protocol capable of reaching an indisputable agreement between two nodes communicating over an unreliable channel. The experimental evaluation carried out in

this work showed that the proposed three-way protocol represents the best trade-off between accuracy and communication complexity, thus no further acknowledge message is convenient. Finding the right trade-off is also relevant considering that the devices installed in vehicles can have limited resources [36, 15].

VPP defines the structure of a unique message, used to implement all the messages of the three-way protocol. The message contains the following fields:

- **ID**: random message identifier;
- **ACK**: acknowledgment value equal to the identifier of the received message, or equal to 0 for the first message;
- **Payload Length**: an integer which represents the number of bytes of the payload field;
- **Payload**: Variable-length field that contains information required for the specific population protocol. Typically, it contains the state of the sender node.

The behavior of the VPP agent can be modeled through the finite state machine shown in Fig. 4.5.

4.3.3 The Adaptive Communication Controller

The *Adaptive Communication Module* strictly works with the OBU which aim, at this layer, is to physically implement the communication protocols, such as WAVE, IEEE 802.11p, and the entire protocol stack adopted for VANETs. In particular, the ACM represents a controller that limits OBU's connections basing on the network conditions: for example, if the network is overloaded because of the high number of vehicles, the module will make sure that nodes decrease the *Sending Rate*, $S_R(t)$. Specifically, the policy of the *adaptive* communications depends on a temporal series of observations made by the vehicle at the current time t. To ensure real-time analysis, the module process these observations within certain time windows in order to extract the required information to adjust $S_R(t)$. We define an observation, O_t, as a set of parameters characterizing the vehicle's

context at the current time t, such as the number of received messages. Assuming the ACM processes observations in a time period Δ_t, data captured within this interval is processed into fixed-length windows of Δ_t observations, i.e., $W(\Delta_t) = \{O_0, \ldots, O_t\}$. In particular, in the proposed approach, each observation represents the number of messages received by the vehicle and the analysis is based on the total number of messages received in the entire observation window. Choosing the proper length for the window is essential because of the impact it could have on the whole system. Short windows can degrade communication performance, in terms of adaptation to network conditions (because the node makes its choices based on punctual observations that may not be representative of actual network conditions), but may speed up the node to correctly set its own parameters. Vice versa, long windows may improve the system performances since more information about incoming communications can be used. Experimental results using windows of different length, suggested us to use fixed-width windows of 10 seconds with overlap.

During the observation, in our system, the vehicle collects the received messages and calculates the receiving rate as the ratio between the number of received messages (considering at most one message per vehicle) and the observation time. Based on the value of the receiving rate computed at a specific time instant, the vehicle will appropriately adapt its sending rate (as number of sent messages per second). The high-level goal of the ACM is that, the changes carried out on the behavior of the single vehicle, will affect the whole network: in fact, all the nodes of the network, even if with different times, will adapt to the network conditions.

Specifically, the goal is that by modulating the sending rate snd_rate, the receiving rate rcv_rate is kept between the thresholds min_rcv_rate and max_rcv_rate. The values of the thresholds have been chosen empirically by performing numerous simulations to vary parameters such as extent and structure of the map, number of vehicles, vehicular density. Specifically, in our case we considered $min\_rcv\_rate = 5pkts/s$ and $max\_rcv\_rate = 30pkts/s$.

The sending rate snd_rate is modified according to a logic of multiplicative increasing and decreasing. Specifically, we consider the following situations:

Algorithm 2: Adaptive Communication Module - Sending Rate update function

```
1 Function updateRates(min_rate, max_rate, w)
2     rcv_rate = w.groupBy(pkt.id).sum()/w.observationTime
3     if rcv_rate > max_rcv_rate then
4         snd_rate = max(min_snd_rate, snd_rate/2)
5     else if rcv_rate < min_rcv_rate then
6         snd_rate = min(max_snd_rate, snd_rate*2)
```

- `rcv_rate > max_rcv_rate`: this condition indicates that, from the node's point of view, there is high network traffic. The node will decrease its sending rate as $snd\_rate \leftarrow snd\_rate/2$

- `rcv_rate < min_rcv_rate`: this condition, instead, could indicate that there is a low number of nodes in the network or, in general, nodes that exchange few messages. In this case, to increase the interactions and information exchanged between nodes, the node will increase its sending rate as $snd\_rate \leftarrow snd\_rate * 2$

The pseudocode of the sending rate update function is shown in Algorithm 2.

Figure 4.7 shows an example of the strategy adopted by the ACM. The thresholds `min_rcv_rate` and `max_rcv_rate` are represented, within which the receive rate `rcv_rate` should be maintained as a result of the changes performed on the sending rate `snd_rate`. For each time instant, we assume that initially the node evaluates the receiving rate, and only then does it change its sending rate.

As it can be seen, for example, at time instant t_1, the node detects `rcv_rate` between `min_rcv_rate` and `max_rcv_rate`. Therefore, it will not change its `snd_rate`. Instead, at time instant t_2, the node detects `rcv_rate` higher than `max_rcv_rate`. In this case, it will update its `snd_rate` by halving it from the previous time instant. The effects of this action will probably be reflected on the next time instant t_3, in which the receiving rate `rcv_rate` will not increase compared to the previous time instant, but will remain constant. Nevertheless, since `rcv_rate` is still higher than `max_rcv_rate`, the node will continue to modify its `snd_rate`, halving it. From the next instant, the detected `rcv_rate` will decrease,

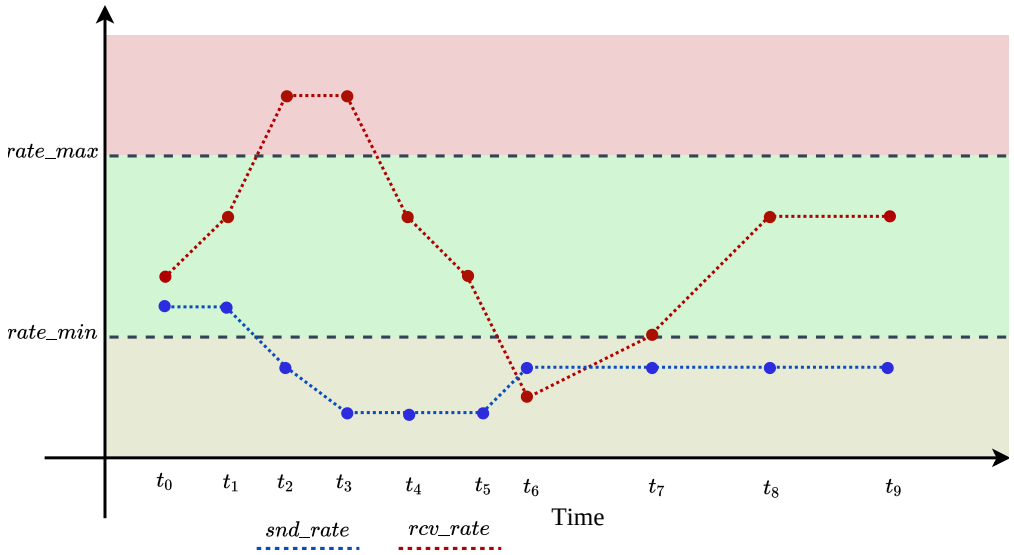


Figure 4.7: Example of how the receiving rate (`rcv_rate`) and sending rate (`snd_rate`) change as a result of the ACM strategy.

until time instant t_6, when it will be even lower than the `min_rcv_rate`. In this case, the node will double its `snd_rate`. From this point on, the `rcv_rate` will always stay within the two thresholds, and therefore the node will no longer change its `snd_rate`.

Clearly this is just an example, but the experimental evaluation shows how effectively the ACM achieves these goals in the different considered scenarios. In any case, thanks to the modularity of the 3-layer architecture, the algorithm implemented by the ACM can be replaced in an immediate and transparent way, in order to adopt a different rate updating strategy.

4.4 Sensing Layer

Fig. 4.8 shows the sensing Layer of the system, which models the set of devices with which the node is equipped and represents the interface with the surrounding world. The main function is to collect information from the sensors and send it to the Data Fusion Algorithm. The raw information from the sensors will be analyzed and processed with the goal of producing the Low-Level Events, which will later be consumed by the upper layer. The Sensing Layer (shown in Fig. 4.8,

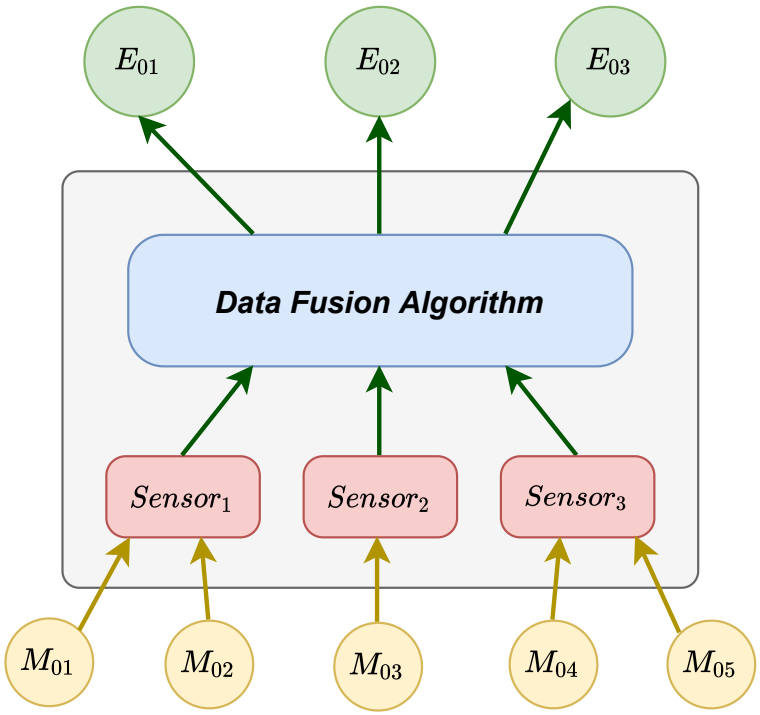


Figure 4.8: Sensing Layer

represents the interface between the node and the surrounding environment. In any case, since it is not the focus of this thesis work, it is not appropriate to go into implementation details.

Chapter 5

Experimental Evaluation

This chapter describes the experimental evaluation and discusses obtained results of the solutions proposed in this work.

Section 5.1 presents the experimental evaluation conducted on the VPP communication schema, describing the scenarios considered, the evaluation metrics adopted and finally the achieved results.

Section 5.2 presents the experimental evaluation of the Event Detection and Dissemination system, describing the experimental setup used, the evaluation metrics adopted, and finally discussing the obtained results.

5.1 VPP - Experimental Evaluation

5.1.1 Impact of Node Mobility

Due to node mobility, errors, and interferences in wireless communications, as previously noted, some inconsistent state transitions can occur. A state transition is defined as inconsistent if the state update function is performed only by a single node, rather than by a pair of nodes.

In wireless communications, the *Packet Error Rate* (PER) is defined as the percentage of packets that are corrupted during a transmission [45]. The expected

value of PER represents the *Packet Error Probability* for a packet M and is defined as:

$$p_p(M) = 1 - (1 - p_e)^{|M|} = 1 - e^{|M|log(1-p_e)}, \tag{5.1}$$

where $|M|$ is the number of bit of M, and p_e is the *bit error probability*. The value of p_e depends on the characteristics of the communication channel, such as noise or distortion.

Another problem influencing VPP is the physical arrangement of nodes. The distance d between two vehicles affects packet transmission due to signal attenuation. To provide a general model, we introduce the function $\gamma(d)$, which models the probability at least one bit of the message is corrupted, during a transmission occurring at a distance d. A suitable distance for transmission can range between 0 (ideally) and R (the communication range of the OBU). The distance varies within the following range:

$$\begin{gathered} \gamma(d) \in [\gamma_{min}; \gamma_{max}] \subseteq [0; 1]\,, \\ \text{where } \gamma_{min} = \gamma(0) \text{ and } \gamma_{max} = \gamma(R). \end{gathered} \tag{5.2}$$

Thus, the probability of having a correct transmission has also to take into account the probability of not observing a message corruption due to the distance between the vehicles, i.e. $(1 - \gamma(d))$.

Finally, a third factor that could affect the transmission is the specific probability of faults due to hardware malfunctions. It is worth noticing that different vehicles can have different probabilities of hardware failure. If φ_x is the probability that a hardware fault will occur during the transmission of a message performed by the vehicle x, the probability of having a correct transmission has also to take into account the factor $(1 - \varphi_x)$.

By combining these factors, we can model the probability that a packet M, sent by the vehicle x, does not reach its destination correctly, as follows:

$$P_\epsilon(M, x) = 1 - (1 - p_p(M))(1 - \gamma(d))(1 - \varphi_x). \tag{5.3}$$

As described above, the only condition in which the system performs an inconsistent transition occurs when the vehicles A and B successfully exchange the messages $M1$ and $M2$, but an error occurs during the transmission of the message $M3$. The probability of such an event occurring can be expressed as follows:

$$P_{inc}(A,B) = \begin{cases} P_\epsilon(M_3, A) \text{ if } A.randId > B.randId \\ P_\epsilon(M_3, B) \text{ otherwhise} \end{cases} \tag{5.4}$$

where

$$\begin{cases} P_\epsilon(M_3, A) = P_\epsilon(\neg M_1, A) \cdot P_\epsilon(\neg M_2, B) \cdot P_\epsilon(M_3, A) \\ P_\epsilon(M_3, B) = P_\epsilon(\neg M_1, B) \cdot P_\epsilon(\neg M_2, A) \cdot P_\epsilon(M_3, B) \end{cases} \tag{5.5}$$

Some further considerations can be made about how the distance between a pair of vehicles varies. We can model their mutual movement, by assuming that for short time interval nodes move along their joining line. Let d_1, d_2 and d_3 be the values of the distance between the pair of nodes when M_1, M_2 and M_3 are transmitted. If vehicles move in the same direction, for a small time interval, we can assume that their distance is constant, i.e., $d_1 \approx d_2 \approx d_3$. If, on the other hand, the vehicles move in opposite directions, even at low speeds, their mutual distance varies during the exchange of messages, i.e. $d_1 \neq d_2 \neq d_3$. This difference can affect the probability of missing the last message, thus making an inconsistent transition.

Let d_0 be the distance between a pair of vehicles, A and B, at the time of first contact. Their position along the joining line depends on their speed, i.e., v_A and v_B. If we consider the position of A at the time of first contact as the origin of a relative reference system, A and B positions are expressed as follows:

$$\begin{cases} x_A(t) = \ \ v_A \cdot t \\ x_B(t) = -\, v_B \cdot t + d_0 \end{cases} \tag{5.6}$$

Consequently, their relative distance can be expressed as follows:

$$\begin{aligned} d(t) &= |x_A(t) - x_B(t)| = \\ &= |v_A \cdot t + v_B \cdot t - d_0| = \\ &= |(v_A + v_B) \cdot t + d_0| \end{aligned} \tag{5.7}$$

If we assume that the distance d_0 also represents the limit distance beyond which the two vehicles will no longer be able to communicate, the total contact time, i.e., the time interval during which the two vehicles can communicate, can be expressed as:

$$\Delta T_c = \frac{2d_0}{(v_A + v_B)}. \tag{5.8}$$

To achieve a successful state transition, the three-way message exchange must take less than ΔT_c time. Moreover, the higher the vehicle speeds, the more likely it is to observe inconsistent state transitions, as the time required to perform the three-way message exchange would be close to or greater than ΔT_c.

5.1.2 A VPP Case Study

The VPP communication scheme is exploited for the following case study. It is considered a scenario where N vehicles $\{v_1, \ldots, v_N\}$ are able to collect and share data about their surroundings in order to detect an excessive traffic condition by means of the measured speed thought On-Board-Units (OBUs), i.e., ad-hoc sensors embedded in vehicles or sensors contained in passengers' smart devices, such as GPS [19]. In general, a situation in which vehicles are traveling with low speed is very informative of vehicular traffic in a specific area.

The considered task can be fully accomplished by a population protocol algorithm that aims to know whether vehicles within a circumscribed geographic area are moving with high or low speeds. A potential approach consists in modeling it as a counting problem (refer to Section 5.1.8), which aims to calculate the difference between vehicles belonging to two different classes, i.e. vehicles with high speed and low speed. It is important to note that, in such a composite scenario, the over-traffic information must meet both geographic and time validity, because only vehicles belonging to a restricted geographic area can benefit from knowing

about the event of interest [28]. A straightforward solution to address both issues could be, also here, to leverage on two Population Protocol models running in parallel to the proposed one. In particular, to address the first issue, it is possible to cluster the population of vehicles. In this case, a Population Protocol model requires the knowledge of information related to the geographic position of the vehicle, which can be retrieved by means of a GPS sensor embedded in the OBUs previously mentioned.

Having this in mind, we can assume that the population is divided into clusters and that the algorithm disseminates the relevant information only within the cluster. This assumption allows the experimental evaluation to be focused in a scenario that consists of a single cluster. The extension to a multi-cluster scenario is trivial. It is sufficient that a vehicle is cluster-aware, so that it can ignore information from other clusters that would not be useful. This is easily achieved by extending the state structure to include cluster information.

Regarding the time validity of an event, we adopted the following solution, which allows to partially solve the well-known problem of vehicle synchronization. In particular, it is required that the Population Protocol algorithm to be re-executed at regular intervals. Specifically, the operations performed by the vehicle v_i at each restart are as follows:

- stop of the Population Protocol algorithm;
- acquisition of information from the surrounding environment, through appropriate sensors. Specifically, speed is detected by performing the difference of two consecutive position measurements through GPS.
- use of logical predicates to establish the initialization value σ_i based on the information sensed from the environment and determine its initial state as $s_i = \Omega(\sigma_i)$ through the input mapping function Ω. In our system we use the binary variable *traffic* that assumes $TRUE$ value if the vehicle speed is higher than a specific threshold, $FALSE$ otherwise;
- start of the Population Protocol algorithm and, therefore, the interactions with other vehicles.

This process requires only that vehicles are synchronized in time, with a level of approximation of even tens of seconds, a realistic assumption in several real-world scenarios.

In order to assess the suitability of VPP as a communication schema to adopt in order to perform Population Protocols over VANETs, we compared it with other communication schemata characterized by a different number of messages. In particular, we considered, the following three protocols:

- **Naive Protocol**: no acknowledgement messages are provided; each node broadcasts its state and updates its state when it receive a message from another node; no different roles for nodes involved in a communication are defined.

- **2-way Protocol**: two roles are defined for nodes involved in a communication, i.e., TX and RX, as in VPP. The RX node updates its state when it receives the initial broadcast message from another node, while the TX node updates its state when it receives an acknowledge corresponding to the last message it sent. The messages exchange of the 2-way Protocol is then defined as follows:

$$M1)\ Tx \rightarrow Rx : [rndId\_tx, state\_tx, ack = 0]$$
$$\textit{(RX updates its state)}$$
$$M2)\ Rx \rightarrow Tx : [rndId\_rx, state\_rx, ack = rndId\_tx]$$
$$\textit{(TX updates its state)}$$

- **4-way Protocol**: requires an additional acknowledgement message with respect to VPP, as follows:

$$
\begin{aligned}
&M1)\ Tx \rightarrow Rx : [rndId\_tx, state\_tx, ack = 0] \\
&M2)\ Rx \rightarrow Tx : [rndId\_rx, state\_rx, ack = rndId\_tx] \\
&M3)\ Tx \rightarrow Rx : [ack = randId\_rx] \\
&\qquad\qquad \textit{(RX updates its state)} \\
&M4)\ Rx \rightarrow Tx : [ack = randId\_tx + 1] \\
&\qquad\qquad \textit{(TX updates its state)}
\end{aligned}
$$

The remainder of this section describes a specific algorithm adopted as a case study, the evaluation metrics used to perform our comparative analysis, specifies the experimental settings and finally described the achieved results.

5.1.3 Considered use-case

The Counting Problem is a specific case of the network-size estimation problem and represents one of the most relevant issues in opportunistic networks.

The counting algorithm presented in [50] is considered as a case study to assess the performance of the VPP communication schema. Authors of [50] model a counting algorithm through a Population Protocol and formally demonstrate its properties and convergence time.

According to this model, the population consists of N agents. Each agent is initialized with an input symbol $\sigma \in \{A, B\}$. The algorithm goal is to estimate the value $k = N_A - N_B$, where N_A is the number of nodes initialized with A, and N_B is the number of nodes initialized with B.

Node initialized with A set their initial state with a positive number M, while nodes initialized with A set their initial state with the negative number $-M$. This protocol updates the state of two interacting nodes with the average of their state values before the transition.

Parameter	**Value**
Input Alphabet	$\Sigma = \{A, B\}$
Input mapping function	$\lambda(\sigma) = \begin{cases} M & \text{if } \sigma = A \\ -M & \text{if } \sigma = B \end{cases}$
Set of states	$S = \{-M, -M+1, ..., M-1, M\}$
Output mapping function	$\Omega(x) = \frac{nx}{m} + \frac{1}{2}$
Output Alphabet	$Z = \{-n, -n+1, ..., n-1, n\}$
Transition function	$f(a,b) = \begin{cases} (\frac{a+b}{2}, \frac{a+b}{2}) & \text{if } a+b \text{ is even} \\ (\frac{a+b-1}{2}, \frac{a+b+1}{2}) & \text{if } a+b \text{ is odd} \end{cases}$

Table 5.1: Paremeters of the Population Protocol which models the counting algorithm.

Formally, the algorithm is characterized by the parameters described in Table 5.1.

5.1.4 Evaluation Metrics

In order to evaluate the performance of different communication protocols, we propose to adopt a set of metrics which are independent of the specific implemented algorithm and a set of metrics which rather depend on it. The latter set of metrics has to be defined for each different considered algorithm.

We adopt the following algorithm-independent metrics:

- **Mean Absolute Percentage Error (MAPE)**: defined as the normalized average of the absolute error made by nodes. If x is the correct value that should be produced by the algorithm and x_i is the output value from the i node, the MAPE is defined as follows:

$$MAPE = \frac{\sum_{i=0}^{N} |x - x_i|}{x \cdot N} \tag{5.9}$$

- **Mean Square Error (MSE)**: defined as the average of square error made by nodes:

$$MSE = \frac{\sum_{i=0}^{N} (x - x_i)^2}{N} \tag{5.10}$$

- **Root Mean Square Error (RMSE)**: defined as follows:

$$RMSE = \sqrt{MSE} \tag{5.11}$$

- **Number of Packets**: the total amount of packets transmitted during an experimental run.

The algorithmic-dependent metric we propose to adopt is related to an invariant property of the Population Protocol considered as a case study. According to the transition function shown in Tab. 5.1, it is possible to prove that the sum of node states remains constant during the whole protocol execution. That is, for any time t, the following equation holds:

$$\sum_{i=1}^{N} node_i.state(0) = \sum_{i=1}^{N} node_i.state(t) \tag{5.12}$$

If some inconsistent state updates are performed (i.e., performed by a single node of an interacting node pair), the eq. 5.12 will not be satisfied. Thus, we adopt the following error function:

$$\xi(t) = \left| \sum_{i=1}^{N} node_i.state(0) - \sum_{i=1}^{N} node_i.state(t) \right| \tag{5.13}$$

5.1.5 Tools used for experimental evaluation

The performance evaluation has been conducted through simulation, by adopting some well-known simulation tools, such as the SUMO/VEINS simulator and the OMNET++ libraries. SUMO is an open source traffic simulator for creating detailed road scenarios, modeling vehicle behavior, and analyzing traffic performance. It offers a wide range of features, including traffic generation algorithms,

vehicle behavior models, road infrastructure modeling, and tools for analyzing results. Thus, SUMO is not merely a traffic simulator, but a suite of applications that allow the definition of everything needed to build an experimental setting for traffic simulations. SUMO is a microscopic traffic simulator, i.e., it allows full control over each vehicle; in fact, each vehicle is explicitly defined, identified by an identifier, departure time, and vehicle path within the network. If desired, each vehicle can be defined with a very high level of detail. Vehicles moving within the map regulate their speed based on the mobility model adopted; SUMO uses an extension of the car-following model developed by Stefan Krauß [41]. This model describes the behavior of moving vehicles when they are one behind the other. The model focuses on studying the movement of vehicles under heavy traffic conditions, where the presence of other vehicles can affect driving behavior, and is based on the notion of headway (distance between the leading vehicle and the following vehicle). The speed of the leading vehicle, the speed of the following vehicle and the speed difference between the two vehicles are used as parameters. In summary, this model is based on some fundamental assumptions:

- **One-dimensional movement:** The model considers only the longitudinal movement of vehicles along a straight line.
- **Rational drivers:** Drivers are assumed to act rationally, trying to maintain an adequate safety distance and achieve a desired speed.
- **Constant speed of leading vehicle:** It is assumed that the vehicle in front maintains a constant speed.
- **Immediate reactions:** Drivers are assumed to react immediately to changes in the traffic situation.
- **Safe distance:** The model considers the importance of maintaining a safe distance between vehicles.
- **Deterministic factors:** The model assumes that drivers' behavior and interactions between vehicles are deterministic.

VEINS is a simulation framework based on SUMO (Simulation of Urban MObility) that focuses on the simulation of Vehicular Ad-Hoc Networks (VANETs). It

is an extension of SUMO that adds vehicular communication capabilities to traffic simulations, allowing the performance of VANET protocols and applications to be evaluated in a realistic environment. In brief, VEINS has the following features:

- **Vehicular communication:** VEINS integrates vehicular communication into SUMO models, allowing vehicles to communicate with each other via vehicular communication technologies, such as DSRC (Dedicated Short Range Communications) or IEEE 802.11p. This communication can be used to exchange traffic information, report accidents, coordinate vehicle movement, and support vehicular safety applications.

- **Signal propagation models:** VEINS provides accurate models for radio signal propagation within VANET simulations. These models consider the effects of terrain, obstacles, and interference on vehicular communications. This makes it possible to evaluate the reliability and coverage of vehicular communications under different traffic scenarios.

- **Support for communication protocols:** VEINS supports several vehicular communication protocols, such as the AODV (Ad-hoc On-Demand Distance Vector) routing protocol, the IEEE 802.11p MAC channel access protocol, and the CAM (Cooperative Awareness Message) and DENM (Decentralized Environmental Notification Message) safety message exchange protocol defined by the European ITS-G5 standard. This allows the effectiveness of these protocols to be evaluated in VANET simulations.

- **Vehicular application development:** VEINS provides an interface for the development of custom vehicular applications. Developers can create applications that take advantage of simulated vehicular information and communications, such as intelligent navigation systems, collision warning systems, traffic management systems, and many others. The effectiveness of such applications can be implemented and evaluated in realistic traffic scenarios.

- **Integration with external tools:** VEINS can be integrated with external tools such as OMNeT++ (Objective Modular Network Testbed in C++) to provide additional simulation and analysis capabilities. Integration with

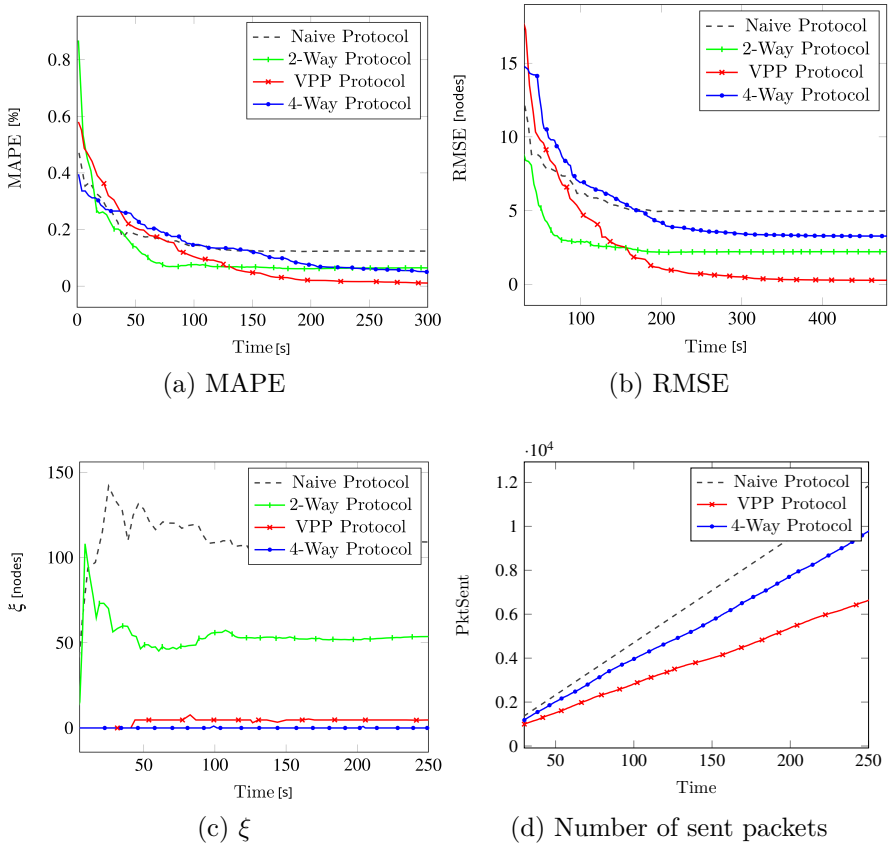
OMNeT++ allows more detailed network and communication aspects to be modeled, enabling even more in-depth performance evaluation of vehicular protocols and applications.

Overall, VEINS provides a comprehensive and flexible simulation environment for studying and evaluating vehicle networks.

5.1.6 Experimental Settings

Each simulation is specified by the following parameters:

- **Map**: a synthetic map composed by a network of streets. We consider two different sizes for maps:
  - **Big Map**: a grid of 16x16 streets, which intersect each other, with a distance of 100 metres between two consecutive parallel streets. The map covers an area of 2.25 km^2, with 25.6 km of roads.
  - **Small Map**: a grid of 8x8 streets, which intersect each other, with a distance of 100 metres between two consecutive parallel streets. The map covers an area of 0.49 km^2, with 6.4 km of roads.
- **Vehicle density**: during the simulation, we considered two different density levels:
  - **High Density**: 31 vehicles per linear kilometer;
  - **Low Density**: 8 vehicles per linear kilometer;
- **Beacon Interval [s]**: Time interval between two consecutive message broadcasting. In our experiments, this value is set to 1s, which is the default value of the used VANETs simulator.
- **Communication range [m]**: Communication range of nodes. Generally between 50 and 80 metres. For our simulation this value is 70m.



(a) MAPE (b) RMSE

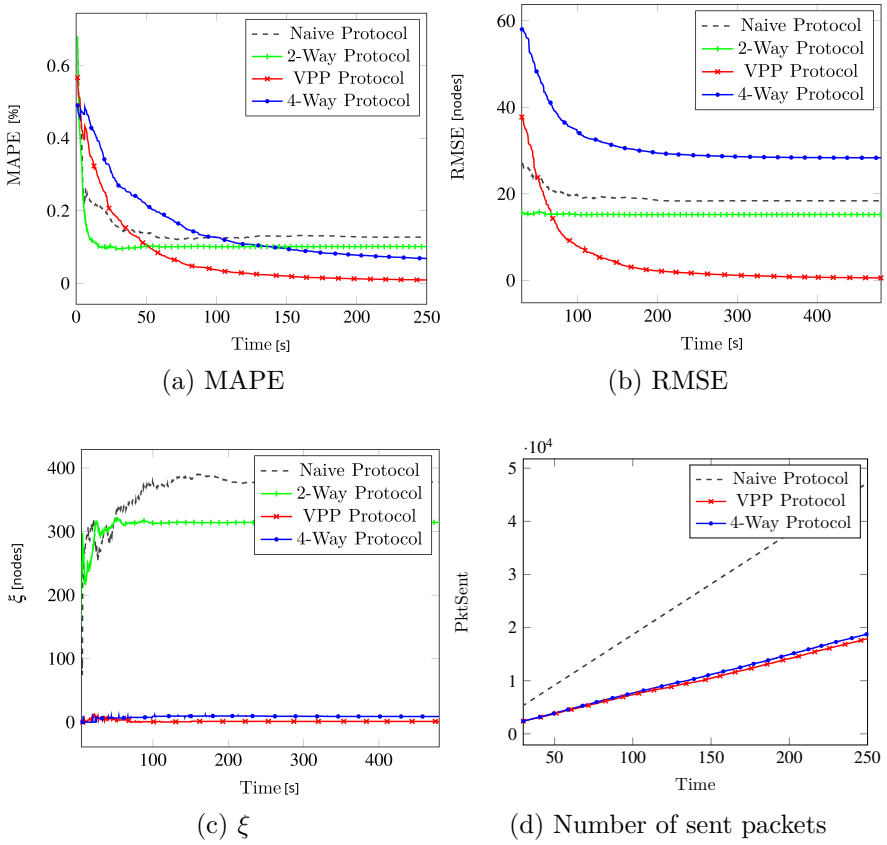
(c) ξ (d) Number of sent packets

Figure 5.1: Comparison of the Mean Absolute Percentage Error (MAPE) (a), of the Root Mean Square Error (RMSE) (b), of the specific error function ξ (c) and the number of packets sent (d), for the four analyzed protocols, in a *Low-Density* scenario over a *Small Map*.

5.1.7 Experimental Results

Experimental results show the suitability of VPP for implementing Population Protocols in VANET, in different scenarios.

Figure 5.1 allows to compare the four analyzed protocol through their Mean Absolute Percentage Error (MAPE), Root Mean Square Error (RMSE), specific error function ξ and the number of packets sent in a *Low-Density* scenario over a *Small Map*. VPP achieves the smallest MAPE and RMSE both with respect to the naive protocol and to the 2-way protocol, that is an expected result, but also with respect to the 4-way protocol. This counterintuitive result is due to



(a) MAPE (b) RMSE

(c) ξ (d) Number of sent packets

Figure 5.2: Comparison of the Mean Absolute Percentage Error (MAPE) (a), of the Root Mean Square Error (RMSE) (b), of the specific error function ξ (c) and of the number of packets sent (d), for the four analyzed protocols, in a *High-Density* scenario over a *Small Map*.

the excessive time span of the 4-way protocol if compared with vehicles time of contact. Such feature causes a greater number of inconsistent transitions with respect to VPP. By considering the algorithm-dependent error function ξ, VPP and the 4-way protocol obtain comparable results, and outperform both the naive protocol and the 2-way protocol.

Figure 5.2 shows the same results, obtained in the same small map, but considering a high density of vehicles. Even in this case VPP outperforms other protocols, especially with respect the MAPE and the RMSE. We can observe comparable results also in *Low-Density* and *High-Density* scenarios over a *Big Map*, respectively reported in Fig. 5.3 and Fig. 5.4.

Experimental results also show that VPP allows to reduce the number of transmitted packet. Figures 5.1d, 5.2d, 5.3d, 5.4d show the number of transmitted packets for the 2-way protocol, for VPP, for the naive protocol and for the 4-way protocol, in the different considered scenarios.

The smaller number of packets transmitted by VPP with respect to the 4-way protocol is an expected result. It is worth noticing that the naive protocol and the 2-Way protocol (which is not shown since it has performance perfectly superimposable to the naive protocol) do not include any mechanism that stops the broadcasting phase. This mechanism is instead included in VPP and in the 4-way protocol. Thus the former pair of protocols obtains worst performance than the latter.

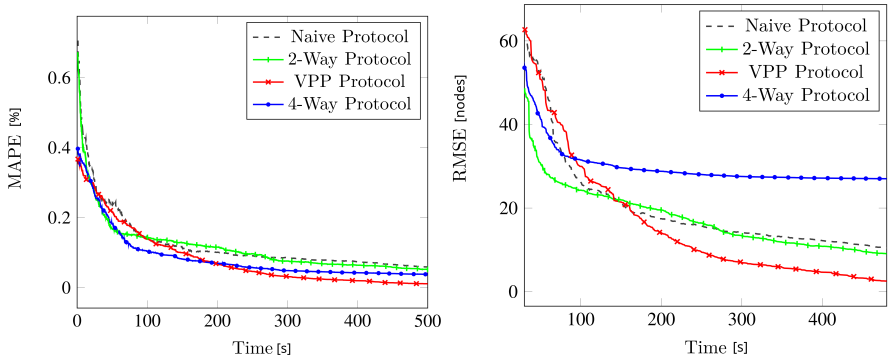
The obtained results thus show that, in three out of four scenarios, VPP obtains the best results both with respect to the performed error and the amount of transmitted messages. In the only scenario in which the 4-way protocol produces a smaller number of messages, its performance in terms of MAPE and RMSE are significantly worst than VPP, thus VPP is still largely preferable.

5.1.8 Case Study Evaluation

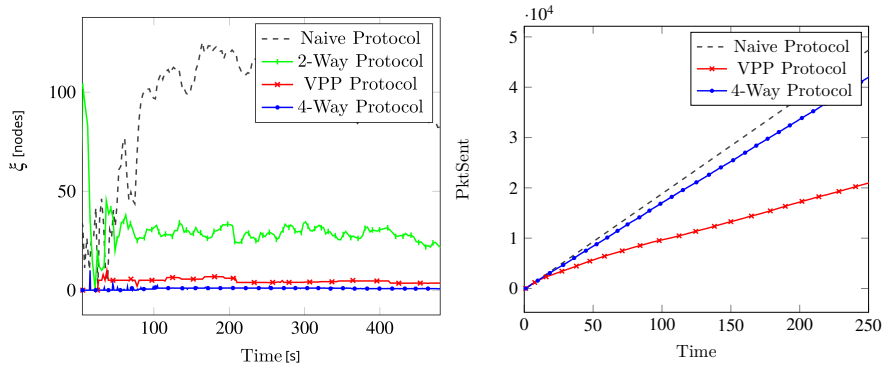
In order to strengthen the experimental evaluation of the proposed scheme, experimental results obtained by testing VPP on the case study described in section 5.1.2 are also reported.

The same evaluation metrics are used, as well as the same configuration of experiments (in terms of parameters). Again, VPP is compared with the other 3 protocols considered previously: Naive-Protocol, 2-way Protocol and 4-way Protocol.

The experimental evaluation of the proposed system provides analysis from both efficiency and accuracy perspectives. Performances related to the efficiency of the protocol were analyzed considering the amount of information exchanged between vehicles. It would be suitable that the protocol requires the exchange of a reduced number of messages to achieve convergence. In this regard, as can be seen from the Fig. 5.5c, the *4-way* schema achieves better results than other protocols. This is due to the fact that, in both the *3-way* and *4-way* protocols,

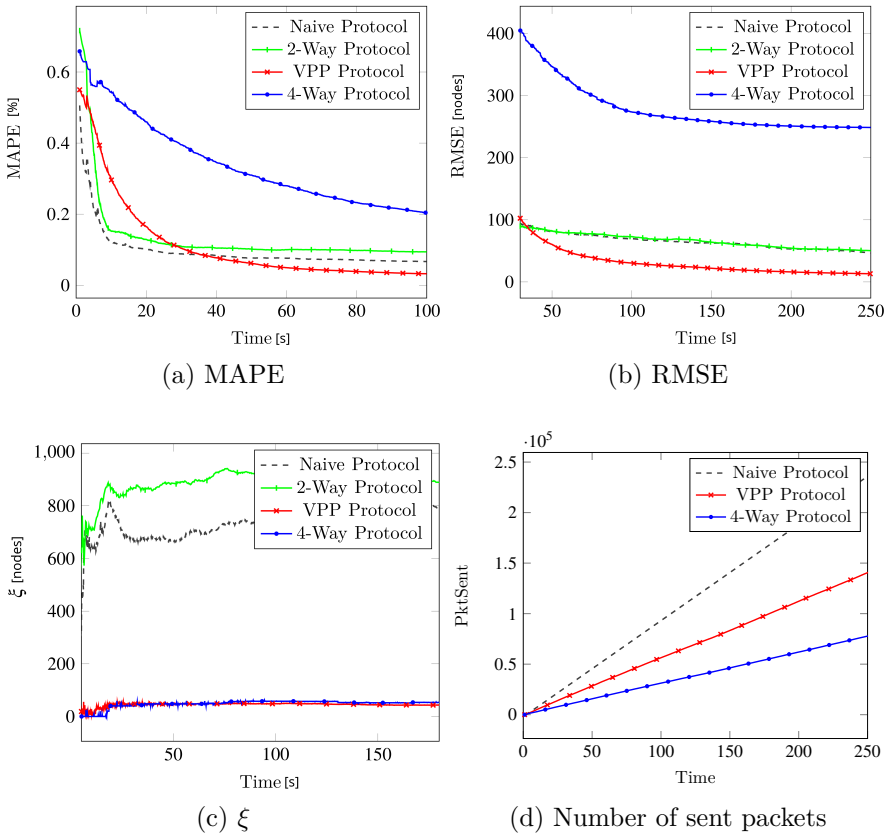


(a) MAPE (b) RMSE



(c) ξ (d) Number of sent packets

Figure 5.3: Comparison of the Mean Absolute Percentage Error (MAPE) (a), of the Root Mean Square Error (RMSE) (b), of the specific error function ξ (c) and of the number of packets sent (d), for the four analyzed protocols, in a *Low-Density* scenario over a *Big Map*.



(a) MAPE (b) RMSE

(c) ξ (d) Number of sent packets

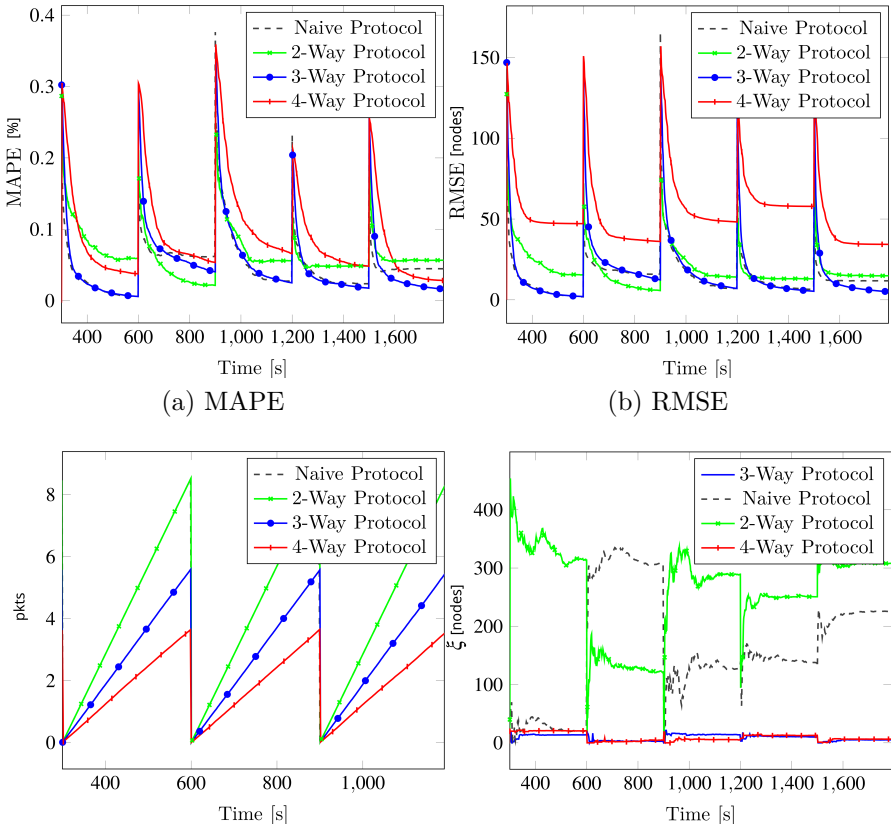
Figure 5.4: Comparison of the Mean Absolute Percentage Error (MAPE) (a), of the Root Mean Square Error (RMSE) (b), of the specific error function ξ (c) and of the number of packets sent (d), for the four analyzed protocols, in a *High-Density* scenario over a *Big Map*.

once the vehicles start communicating (exchanging messages), the vehicles stop the broadcast process, thus reducing the number of sent messages. Since the *4-way* version involves the exchange of 4 messages (M_1, M_2, M_3, and M_4), the total time of interaction between vehicles will be longer and vehicles will spend more time without broadcasting messages. In fact, the *Naive* and *2-way* versions, which do not include any of these mechanisms, exchange even more messages, since the vehicles will always be in the broadcast phase. However, this has a significant impact on the convergence time.

In this regard, Fig. 5.5a, which allows to compare the Mean Absolute Percentage Error ($MAPE$) of the considered protocols, shows that the *3-way* protocol, performing interactions in a faster way, converges to a lower error value faster than the *4-way* version. Instead, as expected, the *Naive* and *2-way* versions, not implementing any type of control on the correct execution of the interaction, cause a greater number of incorrect transitions, resulting in the inconsistency of the system and producing incorrect output values. This is an expected result since the *Naive* and *2-way* versions do not implement mechanisms to ensure that the transition is performed correctly, as is done by the *3-way* protocol.

As for the specific error function ξ, in a theoretical case, it should always remain equal to 0. However, as can be seen from Fig. 5.5d, the ξ function has values differing from 0. Specifically, in the $Naive$ and *2-way* versions, which do not implement mechanisms to ensure that the interactions are successfully completed, this error has very high values. Instead, the *3-way* and *4-way* versions, which implement different control mechanisms, have significantly lower values, close to 0.

The *Naive* and *2-way* protocols achieve the lowest performance in both efficiency and correctness. Instead, the *3-way* and *4-way* schemata obtain comparable performances: both achieve similar results relative to the algorithm-dependent error function ξ however, while *4-way* obtains better results relative to the number of exchanged messages, *3-way* presents better performances relative to the convergence time. Due to the nature of the scenario and the application requirements, the reduction in the number of messages exchanged does not justify the decrease in convergence time performance. In fact, it is preferable to adopt a protocol that requires a larger amount of information exchanged between vehicles but produces



(a) MAPE (b) RMSE

(c) Total number of Messages (d) $\xi(t)$ Error Function

Figure 5.5: Performance of the considered communication protocols, specifically (a) shows the $MAPE$ values, (b) plots the $RMSE$ values, (c) shows the number of messages exchanged by the vehicles and (d) instead reports the values of the ξ Error Function.

correct results faster. For these reasons, the *3-way* protocol represents the best trade-off between the compared alternatives.

5.2 Event Detection System - Experimental Evaluation

This section describes the experimental evaluation of our proposal for event detection system. The experimental setting is first described, then the performance

evaluation of the data dissemination protocol and of the underlying adaptive communication strategy is described.

5.2.1 Experimental Settings

The experimental evaluation was performed through the VEINS framework [56], which is based on the SUMO traffic simulator [43] and the OMNET++ event simulator [61]. We assessed our approach in different scenarios, obtained by varying the following parameters:

- **Map Size**: we have considered two different sizes of maps:
  - *Large Map*: 12 linear km of roads over an area of 1 km^2;
  - *Small Map*: 6 linear km of roads over an area of 0.25 km^2;
- **Vehicle density**: we have considered two different vehicular densities:
  - *High Density*: 40 vehicles per linear kilometer;
  - *Low Density*: 20 vehicles per linear kilometer;
- **GPS availability**: to compare the performance of our approach with or without GPS information, in some scenarios the vehicles are equipped with GPS sensors, and in other scenarios, this information is not available;
- **Broadcasting Range**: the adaptive communication strategy tunes the broadcasting rate within a certain range; we evaluated the performance of our approach by varying the specific value of this range. In scenarios in which vehicles do not adopt an adaptive communication strategy, the broadcasting rate is equal to 1 message per second.

By combining these parameters we evaluated the performance of our approach in 16 different scenarios.

In all the simulated scenarios, the communication range of each vehicle is 70 m.

5.2.2 Evaluation Metrics

This section presents the evaluation metrics adopted to evaluate the performance of our system.

The effectiveness of the data dissemination protocol can be evaluated by considering, for each vehicle, how many events are actually detected in its range of interest. With this goal in mind, we can provide a suitable definition of four well-known metrics, i.e., *True Positives*, *True Negatives*, *False Positives*, and *False Negatives*.

- **True Positives**: (TP_i) Number of events that the vehicle i considers as relevant and are inside its area of interest;
- **True Negatives**: (TN_i) Number of events that the vehicle i does not consider as relevant and are outside its area of interest;
- **False Negatives**: (FN_i) Number of events that the vehicle i does not consider as relevant but are inside its area of interest;
- **False Positives**: (FP_i) Number of events that the vehicle i considers as relevant but, instead, are outside its area of interest.

Assuming that n vehicles are involved in the simulation, we adopted the following evaluation metrics, which are computed by averaging over all vehicles:

- **Precision** - average portion of truly relevant events among all the events detected by vehicles:
$$P = \frac{1}{n} \sum_{i}^{n} \left(\frac{TP_i}{TP_i + FP_i} \right);$$
- **Accuracy** - average portion of events correctly classified among the events occurring in the considered time interval:
$$A = \frac{1}{n} \sum_{i}^{n} \left(\frac{TP_i + TN_i}{TP_i + FP_i + TN_i + FN_i} \right);$$

- **Recall** - average portion of events detected as relevant among all the events occurring in the area of interest of a vehicle (all the events that a vehicle should detect):

$$R = \frac{1}{n} \sum_{i}^{n} \left(\frac{TP_i}{TP_i + FN_i} \right);$$

- **F1-score** - harmonic mean of precision and recall:

$$F1 = \frac{1}{n} \sum_{i}^{n} \left(\frac{P_i \cdot R_i}{P_i + R_i} \right).$$

The efficiency of the adaptive communication strategy, underlying the data dissemination protocol, depends on the rate at which vehicles send packets in the network. An excessive transmission rate results in resource consumption and may cause an excessive number of collisions in the transmission medium. On the other hand, a low transmission rate may cause vehicles to miss some useful information and thus some relevant events.

- $pkts(t)$: number of messages sent by the vehicles, from the beginning of the simulation until time t, averaged over all vehicles.

- $rate(t)$: average transmission rate of all vehicles, at time t.

5.2.3 Evaluation Results

To evaluate the benefits of key design choices, we compared the behavior of three different implementations of our system. The first version of the system, labeled "GPS", requires vehicles to be equipped with GPS sensors, and thus be able to detect their location and associate coordinates with detected events. In this case, the data broadcasting protocol achieves very good performance, which does not change whether the vehicles adopt a communication strategy with a static broadcasting rate or the adaptive communication strategy. Therefore, for the sake of brevity, results shown in this section refer to the system with GPS sensors and a static broadcasting rate. The second and the third versions of the system, involve the more challenging scenario where vehicles are not equipped with GPS sensors. The second version, labeled "Static", adopts a communication strategy with a static

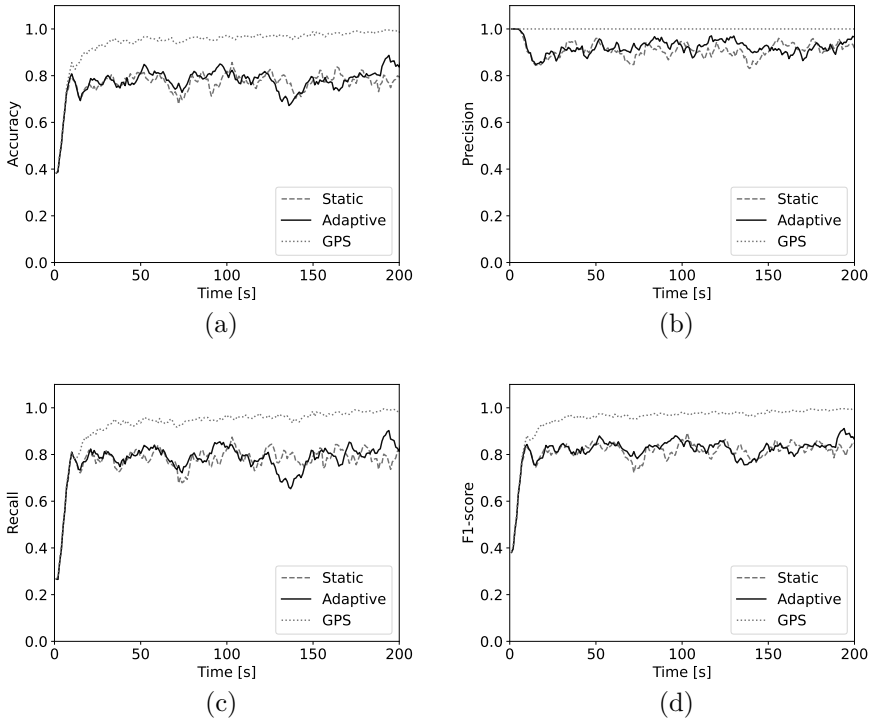


Figure 5.6: Comparison of accuracy 5.6a, precision 5.6b, recall 5.6c and F1-Score 5.6d, in a *small map* with *low density* of vehicles, of three versions of our system: (i) with GPS sensors (labeled "GPS"), (ii) without GPS sensors and with a static broadcasting rate (labeled "Static"), and (iii) without GPS sensors and the adaptive communication strategy (labeled "Adaptive").

broadcasting rate, while the third version, labeled "Adaptive", adopts the adaptive communication strategy.

Figures 5.6 and 5.7 show the performance evaluation of our data dissemination protocol in a *small map*, respectively with a *low* and *high* density of vehicles, by considering these three different versions of our system. Figures 5.8 and 5.9 show the performance evaluation in a *large map*, respectively with a *low* and *high* density of vehicles.

As expected, the system with GPS sensors obtains the best performance, with average values of accuracy, precision, recall and F1-score equal to 0.99, 1.00, 0.98, and 0.98. It can also be observed, that when vehicles are equipped with GPS sensors, the precision is always equal to 1. This is because this system has no false

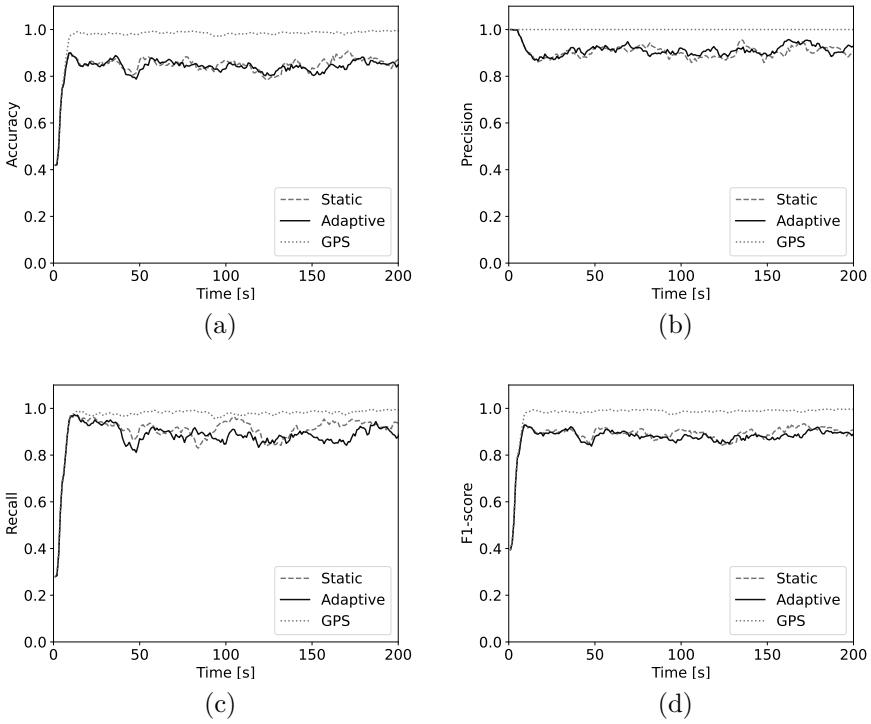


Figure 5.7: Comparison of accuracy 5.7a, precision 5.7b, recall 5.7c and F1-Score 5.7d, in a *small map* with *high density* of vehicles, of three versions of our system: (i) with GPS sensors (labeled "GPS"), (ii) without GPS sensors and with a static broadcasting rate (labeled "Static"), and (iii) without GPS sensors and the adaptive communication strategy (labeled "Adaptive").

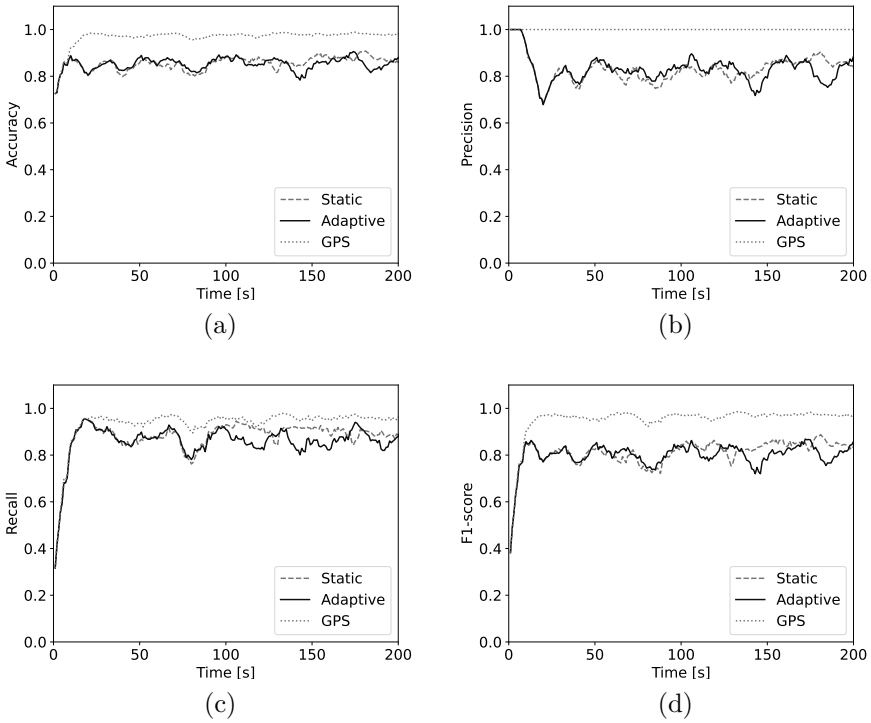


Figure 5.8: Comparison of accuracy 5.7a, precision 5.7b, recall 5.7c and F1-Score 5.7d, in a *large map* with *low density* of vehicles, of three versions of our system: (i) with GPS sensors (labeled "GPS"), (ii) without GPS sensors and with a static broadcasting rate (labeled "Static"), and (iii) without GPS sensors and the adaptive communication strategy (labeled "Adaptive").

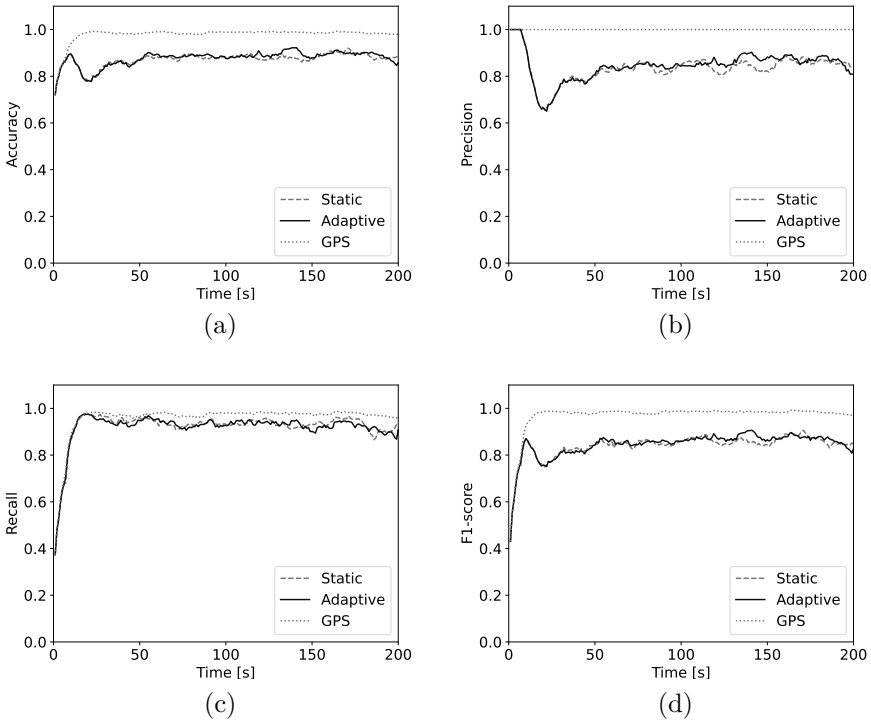
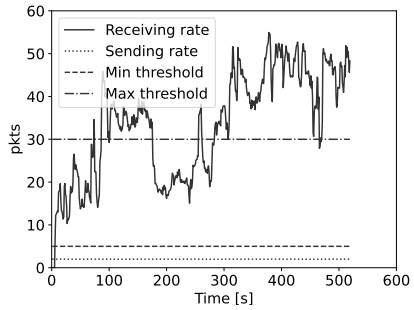


Figure 5.9: Comparison of accuracy 5.7a, precision 5.7b, recall 5.7c and F1-Score 5.7d, in a *large map* with *high density* of vehicles, of three versions of our system: (i) with GPS sensors (labeled "GPS"), (ii) without GPS sensors and with a static broadcasting rate (labeled "Static"), and (iii) without GPS sensors and the adaptive communication strategy (labeled "Adaptive").

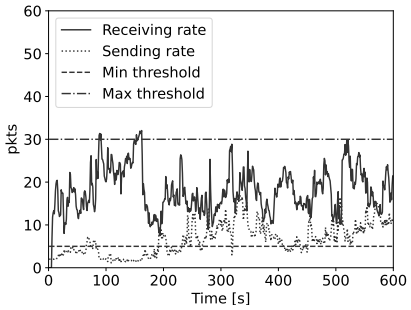
positives. In fact, since the events detected by the sensors are labeled with the GPS coordinates of the place where they have been detected, each node that receives a message is able to correctly detect whether an event is in its area of interest or not. Therefore, it never happens that an event outside its area of interest is mislabeled as relevant.

When the vehicles are not equipped with GPS sensors, our data dissemination protocol achieves average values of accuracy, precision, recall and F1-score equal to 0.85, 0.88, 0.87, and 0.85. It is worth noticing that the adaptive communication strategy, although with a reduced the number of transmitted messages, achieves essentially the same performance as the system adopting the static transmission rate. The system achieves better performance in high-density scenarios, because in denser networks, vehicles exchange more information, and as a result, knowledge of detected events propagates faster.

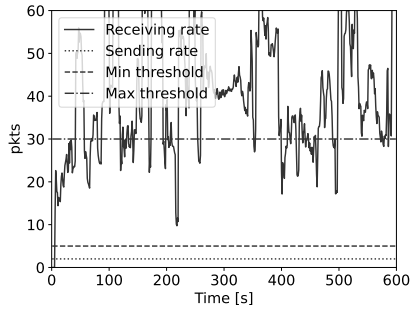
With these results in mind, it is significant to assess the proposed approach, in a scenario where vehicles are not equipped with GPS sensors, the savings achieved by the adaptive communication strategy, which is shown not to adversely affect the performance of the data dissemination protocol. The efficiency of the adaptive communication strategy and that of the data dissemination protocol must be properly balanced. A high transmission rate causes high energy consumption and a relevant number of collisions in the transmission medium. On the other hand, a low transmission rate, which is more convenient from the communication overhead point of view, may cause the vehicles to miss some useful information, which is detrimental to the detection accuracy. Figures 5.10 and 5.11 compare the adaptive communication strategy with the static communication strategy in terms of packet receiving and sending rates. In high-density scenarios (Figure 5.10), the static transmission rate causes an increase in the receive rate in vehicles (Figure 5.10a and 5.10c) due to the high density of vehicles, resulting in a high number of collisions in the transmission medium. It is worth noting that this large communication cost does not correspond to any increase in detection accuracy, as shown above. On the contrary, the adaptive communication strategy (Figure 5.10b and 5.10d), in order to maintain the receiving rate between the minimum and maximum threshold, adaptively tune the senting rate. Consequently, the system causes a lower energy consumption, a lower number of collisions in the transmission medium, without a



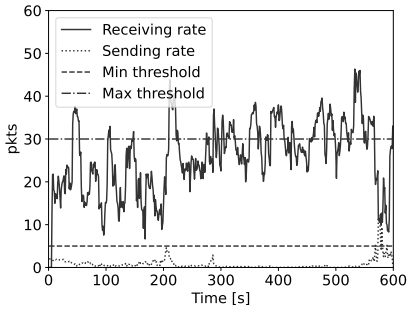
(a) Static communication strategy over a *Large Map*.



(b) Adaptive communication strategy over a *Large Map*.

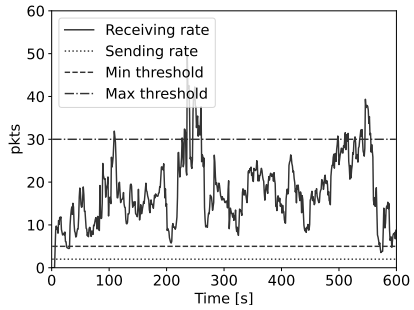


(c) Static communication strategy over a *Small Map*.

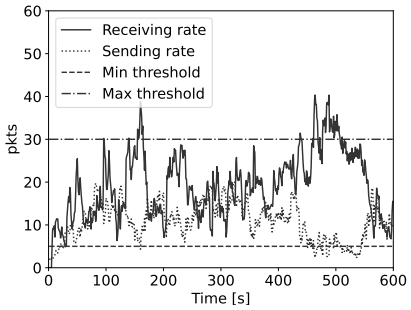


(d) Adaptive communication strategy over a *Small Map*.

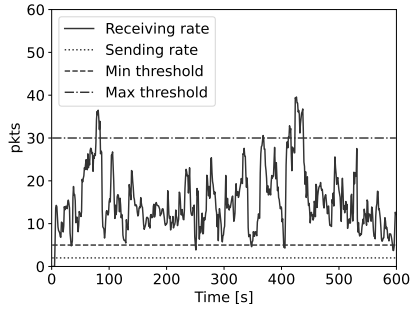
Figure 5.10: Receiving and sending rate in *High Density* scenarios, over *Large Maps* and *Small Maps*, when adopting the static communication strategy (5.10a and 5.10c) and the adaptive one (5.10b and 5.10d).



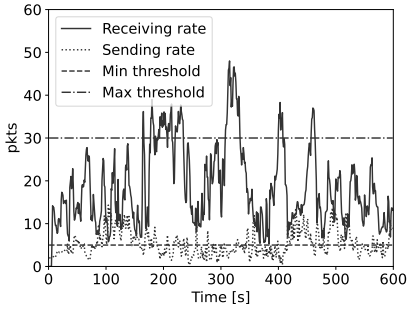
(a) Static communication strategy over a *Large Map*.



(b) Adaptive communication strategy over a *Large Map*.



(c) Static communication strategy over a *Small Map*.



(d) Adaptive communication strategy over a *Small Map*.

Figure 5.11: Receiving and sending rate in *Low Density* scenarios, over *Large Maps* and *Small Map*, of a systema adopting when adopting the static communication strategy (5.11a and 5.11b) and the adaptive one (5.11c and 5.11d).

detrimental effect on the detection accuracy. Instead, as expected, in low-density scenarios the communication burden of the static and adaptive strategies are comparable (see Figure 5.11).

In order to better assess the dependence of energy consumption on the sending rate, Figure 5.12 compares the average number of packets sent by vehicles, which can be considered directly proportional to energy consumption, for the static and the adaptive communication strategies, in all the considered scenarios. It can be shown that in low-density scenarios (Figures 5.12a and 5.12c), the adaptive strategy uses a greater number of sent packets, in order to respond to a potential lack of information, while in high-density scenarios (Figures 5.12b and 5.12d), it limits the number of sent packets, in order to avoid collisions that would not lead to any increase in detection accuracy.

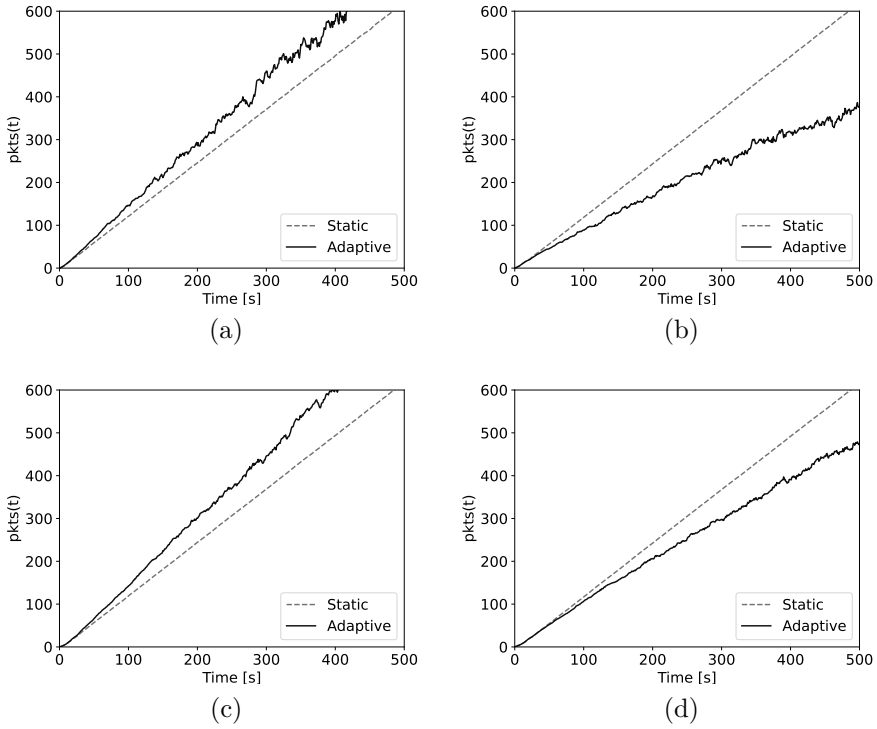


Figure 5.12: Comparison of the number of packets sent by vehicles in *low-density* scenarios over a *Small Map* 5.12a and a *Large Map* 5.12c, and in *high-density* scenarios over a *Small Map* 5.12b and *Large Map* 5.12d in different configuration: (i) with a static broadcasting rate (labeled "Static") and (ii) with the adaptive communication strategy (labeled "Adaptive").

Chapter 6

Conclusions

This Thesis work relates to the emerging field of Vehicular Ad-Hoc Networks (VANET). This area, in recent years has attracted the attention of both industry, interested in the development of innovative applications and business opportunities, and academia, involved in proposing new solutions to critical issues that are still present.

It was discussed how today's VANET architectures can be divided into two categories: Vehicle-To-Vehicle (V2V) and Vehicle-To-Infrastructure (V2I). Although in V2I architectures, vehicles can leverage the computational power of fixed units in the territory (RSUs), which also allow them to integrate with cloud applications, V2I architectures are poorly suited to real-world scenarios due to their high deployment and maintenance costs. Because of these limitations, the scientific community is increasingly pushing the development of applications for V2V networks. These, since the infrastructure is totally distributed and consists only of vehicles interacting with each other, have higher potential. Nevertheless, the adoption of V2V architectures is not trivial, because several challenges must be addressed. One of the main challenges lies in the broader challenge of Data Dissemination, that is disseminating information across the network in an efficient way. The literature has shown that the performance of proposed solutions, regardless of the specific scope of application, is highly dependent on the routing algorithm adopted. In this regard, to date, there is no routing algorithm capable

of dynamically adapting to constantly changing network conditions. In addition, most routing algorithms require an initialization phase, which obviously burdens the efficiency of the overall system.

In this work, with the aim of realizing a system characterized by high performance, the possibility of adopting a different communication model than the conventional routing algorithms was analyzed. It was seen that the Population Protocol model represents a viable alternative to the use of classical communication models. This model represents an elegant formalism for dealing with the typical problems of distributed systems. Although due to its characteristics it completely fulfills the requirements of vehicular networks, its use in a real-world context required solving some problems related to physical constraints that are considered in the theoretical model. Specifically, two problems were encountered in using Population Protocol-based algorithms in a real-world context: the first is to ensure that, despite the possibility of communication errors, interactions between nodes occur in the correct way. The second problem concerns the possibility of executing asymmetric algorithms as well, that is, those that implement a transition function that updates the states of the two interacting nodes in a different way.

To solve these problems, the 3-way communication schema VPP was designed, which, through specific and appropriate mechanisms, enables the use of Population Protocol-based algorithms in vehicular networks. Experimental evaluation of VPP demonstrated the validity of the solution. In fact, it was shown that at the expense of a negligible decrease in performance on the convergence times of the algorithm, VPP succeeds in drastically decreasing the number of erroneous transitions and, therefore, the error occurred on the output value will be significantly lower.

By defining this communication scheme that enables the use of PP in vehicular networks, the full power of PP can be exploited in the Event Detection and Dissemination system proposed in this work.

The Event Detection and Dissemination System is executed in the vehicular network in a fully distributed fashion: it requires no network infrastructure and no initialization phase. It adopts an Event Driven approach, in which components react asynchronously to events that occur. In this regard, two types of events have been defined: Low-Level Events represent simple events, which are produced

locally by individual nodes. High-Level Events are produced through Data Fusion algorithms applied to multiple LLEs. HLEs represent compound events that are consumed directly by the services provided to the user.

The system has a twofold function: on the one hand, it allows nodes, by means of analyses performed on sensory measurements, to determine whether events have occurred in the surrounding environment; on the other hand, it must propagate this information to other nodes in the network as efficiently as possible. In addition, the system is able, through analyses conducted on current network conditions, to independently adjust its message sending rate, pursuing efficiency and fairness goals.

The system is characterized by a 3-layer architecture. The Application Layer implements the highest-level functionality: it integrates with smart environments and provides services to the user according to the information received from the other layers. Specifically, Complex Event Processing (CEP) receives LLEs from the lower layers, processes and aggregates them to produce HLEs, which will then be consumed by the Service Controllers associated with all services provided by the system.

The Communication layer represents the core of the entire system, because it is responsible for both propagating information within the network and adapting the sending rate based on network conditions. It is composed of the Population Protocol Module, which implements the Data Dissemination algorithm based on the Population Protocol model, and the Adaptive Communication Module (ACM), which, based on observations made on the network, is able to modulate the node's sending rate so as to minimize the probability of saturating the network while still feeding an adequate level of information into the network.

The Sensing Layer models the sensory equipment with which the node is equipped. The sensory measurements performed are collected and processed by an appropriate Data Fusion algorithm, producing as output the Low-Level Events that will be sent to the higher layer.

Experimental evaluation was conducted through the Veins software, a framework that integrates both a vehicular traffic simulator and network communication simulation libraries. Different scenarios were defined based on certain parameters: map size, vehicular density, communication ranges, etc. In order to test the effec-

tiveness and advantages introduced by using the Adaptive Communication Module, all scenarios were run in both configurations: either using the ACM or not using it.

The experimental results show the validity of the proposed system in all scenarios, even with very different conditions. It was seen that using the ACM allows to decrease the number of messages sent when the network is very dense, with the same performance on convergence times. The experimental evaluation also demonstrated the validity of the Data Dissemination algorithm implemented through the Population Protocol model. It was seen how efficiently nodes are able to disseminate information within the network, even in non-optimal network conditions.

Bibliography

[1] Imen Achour et al. “SEAD: A Simple and Efficient Adaptive Data Dissemination Protocol in Vehicular Ad-Hoc Networks”. In: *Wirel. Netw.* 22.5 (July 2016), pp. 1673–1683. ISSN: 1022-0038. DOI: 10.1007/s11276-015-1050-9. URL: https://doi.org/10.1007/s11276-015-1050-9.

[2] Vincenzo Agate, Federico Concone, and Pierluca Ferraro. “A Resilient Smart Architecture for Road Surface Condition Monitoring”. In: *Innovations in Smart Cities Applications Volume 5*. Ed. by Mohamed Ben Ahmed et al. Cham: Springer International Publishing, 2022, pp. 199–209. ISBN: 978-3-030-94191-8. DOI: 10.1007/978-3-030-94191-8_16.

[3] Yasuhiro Akagi. “Ontology based collection and analysis of traffic event data for developing intelligent vehicles”. In: *2017 IEEE 6th Global Conference on Consumer Electronics (GCCE)*. IEEE. 2017, pp. 1–5. DOI: 10.1109/GCCE.2017.8229339.

[4] Ranwa Al Mallah, Alejandro Quintero, and Bilal Farooq. “Distributed Classification of Urban Congestion Using VANET”. In: *Trans. Intell. Transport. Syst.* 18.9 (Sept. 2017), pp. 2435–2442. DOI: 10.1109/TITS.2016.2641903. URL: https://doi.org/10.1109/TITS.2016.2641903.

[5] Mahmoud Al Shareeda, Ayman Khalil, and Walid Fahs. “Towards the optimization of road side unit placement using genetic algorithm”. In: *2018 International Arab Conference on Information Technology (ACIT)*. IEEE. 2018, pp. 1–5. DOI: 10.1109/ACIT.2018.8672687.

[6] Dan Alistarh, James Aspnes, and Rati Gelashvili. "Space-optimal majority in population protocols". In: *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2018, pp. 2221–2239. DOI: 10.48550/arXiv.1704.04947.

[7] Dana Angluin et al. "Computation in networks of passively mobile finite-state sensors". In: *Distributed computing* 18.4 (2006), pp. 235–253. DOI: 10.1007/s00446-005-0138-3.

[8] Dana Angluin et al. "The computational power of population protocols". In: *Distributed Computing* 20.4 (2007), pp. 279–304. DOI: 10.48550/arXiv.cs/0608084.

[9] Maen M Artimy, William J Phillips, and William Robertson. "Connectivity with static transmission range in vehicular ad hoc networks". In: *3rd Annual Communication Networks and Services Research Conference (CNSR'05)*. IEEE. 2005, pp. 237–242. DOI: 10.1109/CNSR.2005.29.

[10] James Aspnes and Eric Ruppert. "An introduction to population protocols". In: *Middleware for Network Eccentric and Mobile Applications*. Springer, 2009, pp. 97–120. DOI: 10.1007/978-3-540-89707-1_5.

[11] Petra Berenbrink, Dominik Kaaser, and Tomasz Radzik. "On Counting the Population Size". In: *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*. PODC '19. Toronto ON, Canada: Association for Computing Machinery, 2019, pp. 43–52. ISBN: 9781450362177. DOI: 10.1145/3293611.3331631. URL: https://doi.org/10.1145/3293611.3331631.

[12] Petra Berenbrink et al. *A population protocol for exact majority with $O(\log^{5/3} n)$ stabilization time and asymptotically optimal number of states*. 2018. DOI: 10.48550/arXiv.1805.05157. arXiv: 1805.05157 [cs.DC].

[13] Antonio Bordonaro, Alessandra De Paola, and Giuseppe Lo Re. "VPP: A Communication Schema for Population Protocols in VANET". In: *2021 20th International Conference on Ubiquitous Computing and Communications (IUCC/CIT/DSCI/SmartCNS)*. IEEE. 2021, pp. 11–18. DOI: 10.1109/IUCC-CIT-DSCI-SmartCNS55181.2021.00017.

[14] Antonio Bordonaro et al. “Modeling Efficient and Effective Communications in VANET through Population Protocols”. In: *2021 IEEE International Conference on Smart Computing (SMARTCOMP)*. 2021, pp. 305–310. DOI: 10.1109/SMARTCOMP52413.2021.00064.

[15] Antonio Bordonaro et al. “On-board energy consumption assessment for symbolic execution models on embedded devices”. In: *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. Vol. 1. IEEE. 2020, pp. 1359–1362. DOI: 10.1109/ETFA46521.2020.9211981.

[16] Safdar Hussain Bouk et al. “Hybrid Adaptive Beaconing in Vehicular Ad Hoc Networks: A Survey”. In: *International Journal of Distributed Sensor Networks* 11.5 (2015), p. 390360. DOI: 10.1155/2015/390360.

[17] Moumena Chaqfeh, Abderrahmane Lakas, and Imad Jawhar. “A survey on data dissemination in vehicular ad hoc networks”. In: *Vehicular Communications* 1.4 (2014), pp. 214–225. ISSN: 2214-2096. DOI: https://doi.org/10.1016/j.vehcom.2014.09.001. URL: https://www.sciencedirect.com/science/article/pii/S2214209614000448.

[18] Hsu-Yung Cheng and Shih-Han Hsu. “Intelligent Highway Traffic Surveillance With Self-Diagnosis Abilities”. In: *IEEE Transactions on Intelligent Transportation Systems* 12.4 (2011), pp. 1462–1472. DOI: 10.1109/TITS.2011.2160171.

[19] Federico Concone, Pierluca Ferraro, and Giuseppe Lo Re. “Towards a Smart Campus Through Participatory Sensing”. In: *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*. 2018, pp. 393–398. DOI: 10.1109/SMARTCOMP.2018.00035.

[20] Federico Concone et al. “A fog-assisted system to defend against Sybils in vehicular crowdsourcing”. In: *Pervasive and Mobile Computing* (2022), p. 101612. ISSN: 1574-1192. DOI: 10.1016/j.pmcj.2022.101612.

[21] Federico Concone et al. “A Novel Recruitment Policy to Defend against Sybils in Vehicular Crowdsourcing”. In: *2021 IEEE International Conference*

on Smart Computing (SMARTCOMP). 2021, pp. 105–112. DOI: 10.1109/SMARTCOMP52413.2021.00035.

[22] Gianpaolo Cugola and Alessandro Margara. "TESLA: a formally defined event specification language". In: *Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems*. 2010, pp. 50–61. DOI: 10.1145/1827418.1827427.

[23] Gianpaolo Cugola et al. "Introducing uncertainty in complex event processing: model, implementation, and validation". In: *Computing* 97.2 (2015), pp. 103–144. DOI: 10.1007/s00607-014-0404-y.

[24] Felipe Cunha et al. "Data communication in VANETs: Protocols, applications and challenges". In: *Ad Hoc Networks* 44 (2016), pp. 90–103. DOI: 10.1016/j.adhoc.2016.02.017.

[25] Alessandra De Paola and Luca Gagliano. "Design of an adaptive bayesian system for sensor data fusion". In: *Advances in Intelligent Systems and Computing* 260 (2014), pp. 61–76. DOI: 10.1007/978-3-319-03992-3_5.

[26] Alessandra De Paola and Marco Morana. "Bio-inspired sensory data aggregation". In: *Advances in Intelligent Systems and Computing* 196 AISC (2013), pp. 367–368. DOI: 10.1007/978-3-642-34274-5_63.

[27] Alessandra De Paola et al. "An Adaptive Bayesian System for Context-Aware Data Fusion in Smart Environments". In: *IEEE Transactions on Mobile Computing* 16.6 (2017), pp. 1502–1515.

[28] Alessandra De Paola et al. "Context-Awareness for Multi-sensor Data Fusion in Smart Environments". In: *AI*IA 2016 Advances in Artificial Intelligence*. Cham: Springer International Publishing, 2016, pp. 377–391. ISBN: 978-3-319-49130-1. DOI: 10.1007/978-3-319-49130-1_28.

[29] Alessandra De Paola et al. "Mimicking biological mechanisms for sensory information fusion". In: *Biologically Inspired Cognitive Architectures* 3 (2013), pp. 27–38. DOI: 10.1016/j.bica.2012.09.002.

[30] Alessandra De Paola et al. "Multi-sensor fusion through adaptive bayesian networks". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6934 LNAI (2011), pp. 360–371. DOI: 10.1007/978-3-642-23954-0_33.

[31] A. D. Devangavi and R. Gupta. "Routing protocols in VANET — A survey". In: *2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon)*. IEEE, 2017, pp. 163–167. DOI: 10.1109/SmartTechCon.2017.8358362.

[32] Giuseppe A Di Luna et al. "Population protocols with faulty interactions: the impact of a leader". In: *Theoretical Computer Science* 754 (2019), pp. 35–49. DOI: 10.1016/j.tcs.2018.09.005.

[33] David Doty and David Soloveichik. "Stable leader election in population protocols requires linear time". In: *Distributed Computing* 31.4 (2018), pp. 257–271. DOI: 10.1007/s00446-016-0281-z.

[34] Elias C Eze et al. "Advances in vehicular ad-hoc networks (VANETs): Challenges and road-map for future development". In: *International Journal of Automation and Computing* 13.1 (2016), pp. 1–18. DOI: 10.1007/s11633-015-0913-y.

[35] Andreas Festag. "Cooperative intelligent transport systems standards in europe". In: *IEEE Communications Magazine* 52.12 (2014), pp. 166–172. DOI: 10.1109/MCOM.2014.6979970.

[36] Salvatore Gaglio et al. "Dc4cd: A platform for distributed computing on constrained devices". In: *ACM Transactions on Embedded Computing Systems (TECS)* 17.1 (2017), pp. 1–25. DOI: 10.1145/3105923.

[37] Rachid Guerraoui and Eric Ruppert. *Even small birds are unique: Population protocols with identifiers*. Tech. rep. 2007.

[38] M Milton Joe and B Ramakrishnan. "Review of vehicular ad hoc network communication models including WVANET (Web VANET) model and WVANET future research directions". In: *Wireless networks* 22.7 (2016), pp. 2369–2386. DOI: 10.1007/s11276-015-1104-z.

[39] Tanuja K et al. “Article: A Survey on Vanet Technologies”. In: *International Journal of Computer Applications* 121.18 (July 2015), pp. 1–9. DOI: 10.5120/21637-4965.

[40] Behnam Khazael, Hadi Tabatabaee Malazi, and Siobhán Clarke. “Complex event processing in smart city monitoring applications”. In: *IEEE Access* 9 (2021), pp. 143150–143165. DOI: 10.1109/ACCESS.2021.3119975.

[41] Stefan Krauß. “Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics”. In: (1998).

[42] Shahid Latif et al. “A comparative study of scenario-driven multi-hop broadcast protocols for VANETs”. In: *Vehicular Communications* 12 (2018), pp. 88–109. DOI: 10.1016/j.vehcom.2018.01.009.

[43] Pablo Alvarez Lopez et al. “Microscopic Traffic Simulation using SUMO”. In: *2018 21st international conference on intelligent transportation systems (ITSC)*. IEEE. 2018. DOI: 10.1109/ITSC.2018.8569938.

[44] Xiaomin Ma, Xiaoyan Yin, and Kishor S Trivedi. “On the reliability of safety applications in VANETs”. In: *International Journal of Performability Engineering* 8.2 (2012), p. 115. DOI: 10.23940/ijpe.12.2.p115.mag.

[45] Aamir Mahmood and Riku Jäntti. “Packet error rate analysis of uncoded schemes in block-fading channels using extreme value theory”. In: *IEEE Communications Letters* 21.1 (2016), pp. 208–211. DOI: 10.1109/LCOMM.2016.2615300.

[46] Avleen Kaur Malhi, Shalini Batra, and Husanbir Singh Pannu. “Security of vehicular ad-hoc networks: A comprehensive survey”. In: *Computers & Security* 89 (2020), p. 101664. DOI: 10.1016/j.cose.2019.101664.

[47] Othon Michail, Ioannis Chatzigiannakis, and Paul G Spirakis. “Mediated population protocols”. In: *Theoretical Computer Science* 412.22 (2011), pp. 2434–2450. DOI: 10.1016/j.tcs.2011.02.003.

[48] Othon Michail, Ioannis Chatzigiannakis, and Paul G Spirakis. “New models for population protocols”. In: *Synthesis Lectures on Distributed Computing Theory* 2.1 (2011), pp. 1–156. DOI: 10.1007/978-3-031-02004-9.

[49] George K. Mitropoulos et al. “Wireless Local Danger Warning: Cooperative Foresighted Driving Using Intervehicle Communication”. In: *IEEE Transactions on Intelligent Transportation Systems* 11.3 (2010), pp. 539–553. DOI: 10.1109/TITS.2009.2034839.

[50] Yves Mocquard et al. “Counting with population protocols”. In: *2015 IEEE 14th International Symposium on Network Computing and Applications*. IEEE. 2015. DOI: 10.1109/NCA.2015.35.

[51] Renê Oliveira et al. “Reliable data dissemination protocol for VANET traffic safety applications”. In: *Ad Hoc Networks* 63 (2017), pp. 30–44. ISSN: 1570-8705. DOI: https://doi.org/10.1016/j.adhoc.2017.05.002. URL: https://www.sciencedirect.com/science/article/pii/S1570870517300835.

[52] Mohammad Al-Rabayah and Robert Malaney. “A New Scalable Hybrid Routing Protocol for VANETs”. In: *IEEE Transactions on Vehicular Technology* 61.6 (2012), pp. 2625–2635. DOI: 10.1109/TVT.2012.2198837.

[53] Sami Abduljabbar Rashid et al. “Reliable and efficient data dissemination scheme in VANET: a review”. In: *International Journal of Electrical and Computer Engineering (IJECE)* 10.6 (2020), pp. 6423–6434. DOI: 10.11591/ijece.v10i6.pp6423-6434.

[54] Ryoya Sadano et al. “A Population Protocol Model with Interaction Probability Considering Speeds of Agents”. In: *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. 2019, pp. 2113–2122. DOI: 10.1109/ICDCS.2019.00208.

[55] Hamayoun Shahwani et al. “A comprehensive survey on data dissemination in Vehicular Ad Hoc Networks”. In: *Vehicular Communications* 34 (2022), p. 100420. ISSN: 2214-2096. DOI: https://doi.org/10.1016/j.vehcom.2021.100420. URL: https://www.sciencedirect.com/science/article/pii/S2214209621000899.

[56] Christoph Sommer, Reinhard German, and Falko Dressler. “Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis”.

In: *IEEE Transactions on Mobile Computing* 10.1 (2011), pp. 3–15. DOI: 10.1109/TMC.2010.133.

[57] Yuichi Sudo et al. "Logarithmic expected-time leader election in population protocol model". In: *International Symposium on Stabilizing, Safety, and Security of Distributed Systems*. Springer. 2019, pp. 323–337. DOI: 10.1007/978-3-030-34992-9_26.

[58] Saif Al-Sultan et al. "A comprehensive survey on vehicular ad hoc network". In: *Journal of network and computer applications* 37 (2014), pp. 380–392. DOI: 10.1016/j.jnca.2013.02.036.

[59] Kanitsorn Suriyapaiboonwattana, Chotipat Pornavalai, and Goutam Chakraborty. "An adaptive alert message dissemination protocol for VANET to improve road safety". In: *2009 IEEE International Conference on Fuzzy Systems*. 2009, pp. 1639–1644. DOI: 10.1109/FUZZY.2009.5277261.

[60] Fernando Terroso-Saenz et al. "A Cooperative Approach to Traffic Congestion Detection With Complex Event Processing and VANET". In: *IEEE Transactions on Intelligent Transportation Systems* 13.2 (2012), pp. 914–929. DOI: 10.1109/TITS.2012.2186127.

[61] Andras Varga. "OMNeT++". In: *Modeling and tools for network simulation*. Springer, 2010, pp. 35–59. DOI: 10.1007/978-3-642-12331-3_3.

[62] Rui Wang et al. "EasiTia: A Pervasive Traffic Information Acquisition System Based on Wireless Sensor Networks". In: *IEEE Transactions on Intelligent Transportation Systems* 12.2 (2011), pp. 615–621. DOI: 10.1109/TITS.2010.2096467.

[63] Chuan Xu et al. "Data collection in population protocols with non-uniformly random scheduler". In: *Theoretical Computer Science* 806 (2020), pp. 516–530. DOI: 10.1016/j.tcs.2019.08.029.