

ANÁLISE FORMAL DE REDES VEICULARES
COM VERIFICAÇÃO DE MODELOS
PROBABILÍSTICA

BRUNO FERREIRA

ANÁLISE FORMAL DE REDES VEICULARES
COM VERIFICAÇÃO DE MODELOS
PROBABILÍSTICA

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

ORIENTADOR: SÉRGIO VALE AGUIAR CAMPOS

Belo Horizonte
Setembro de 2016

BRUNO FERREIRA

**VERIFICATION OF VEHICULAR NETWORKS
USING PROBABILISTIC MODEL CHECKING**

Thesis presented to the Graduate Program
in Computer Science of the Federal
University of Minas Gerais in partial
fulfillment of the requirements for the
degree of Doctor in Computer Science.

ADVISOR: SÉRGIO VALE AGUIAR CAMPOS

Belo Horizonte
September 2016

© 2016, Bruno Ferreira.
Todos os direitos reservados.

Ferreira, Bruno

F383v Verification of Vehicular Networks Using
Probabilistic Model Checking / Bruno Ferreira. — Belo
Horizonte, 2016
xxiii, 159 f. : il. ; 29cm

Tese (doutorado) — Federal University of Minas
Gerais

Orientador: Sérgio Vale Aguiar Campos

1. Computação - Teses. 2. Redes veiculares.
3. Mobile ad hoc network. 4. Vehicular ad hoc
networks. 5. Probabilistic model checking.
I. Orientador. II. Título.

CDU 519.6*22(043)




UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO


FOLHA DE APROVAÇÃO

Verification of vehicular networks using probabilistic model checking

BRUNO FERREIRA


Tese defendida e aprovada pela banca examinadora constituída pelos Senhores:


PROF. SÉRGIO VALE AGUIAR CAMPOS - Orientador
Departamento de Ciência da Computação - UFMG


PROF. ANTONIO ALFREDO FERREIRA LOUREIRO
Departamento de Ciência da Computação - UFMG


PROF. JOSÉ MARCOS SILVA NOGUEIRA
Departamento de Ciência da Computação - UFMG


PROFA. LEILA RIBEIRO
Instituto de Informática - UFRGS


PROF. MARK ALAN JUNHO SONG
Departamento de Ciência da Computação - PUC-MG

Belo Horizonte, 09 de setembro de 2016.

Acknowledgments

This work is not only the result of an individual effort. It was born of significant contributions acquired during my professional and academic career and had support from institutions which were essential to this document.

I thank God to provide me strength, health and peace whenever I needed.

Whole family, especially my beloved mother Maria and my father Divino, to my dear brothers Efrem and Viviane, my eternal gratitude. You are simply fundamental in my life, directly responsible for my emotional health.

I appreciate the loving presence, the help and encouragement from Gizelle. To her, my deepest gratitude.

My thanks and my affectionate homage to Sergio Campos. More than my teacher and advisor during Ph.D. degree, thanks for your complicity and direct responsibility in the construction of this thesis.

Aware that it is impossible to list all which in one way or another contributed to this thesis, I must express my gratitude to friends I got and they gave me so much support Fernando Braz, Paulo Batista, Lucas Hanke, Herbert Rausch and Alessandra Campos. I also thank my DINTER colleagues, especially Paloma and Manoel.

Collectively, I need to register the contribution of teams which I participated like Laboratory of Access Universalization (LUAR), the Computing Science Department of IFMG-Campus Formiga and still among the institutions, I extend a special thank to IFMG direction by unconditional support.

I need to honor also the dear friends who have contributed with their strength and encouragement in order to I could complete this goal.

I would like to express my gratitude to the member of my thesis defense committee. J. Marcos (UFMG), A. A. F. Loureiro (UFMG), L. Ribeiro (UFRGS), and M. A. J. Song (PUC-MG).

“What we anticipate seldom occurs; what we least expected generally happens.”
(Benjamin Disraeli)

Abstract

Vehicular Ad-hoc Networks (VANET) are a particular case of mobile ad hoc network (MANET) in which nodes are vehicles that move following more predictable patterns. This network is based on wireless communication among vehicles and fixed controller nodes deployed on roads or streets. There are several application types and a good example is a Virtual Traffic Light (VTL). These semaphores are implemented in a distributed manner by vehicles themselves, thus, the signaling schedule can be adapted with respect to the current traffic volume. It has shown improved results in traffic control, increasing the traffic flow and reducing costs in infrastructure. However, this application and many others in VANET already have great challenges. An interesting question is how they will be tested. Simulation still is widely used in order to minimize costs and reduce time of the experiments, since physical elements are not necessary and tests can be reproduced easily.

The fields of computer networks and traffic engineering make extensive use of simulators. Thus, there are long established software in both fields. Since the introduction of vehicular networks, the integration of these two areas has recently become necessary. This interaction is required due to inherent features of the strong coupling between communication and mobility in VANETs. Applications alter mobility patterns, which in turn, can disturb the data communication among vehicles. Therefore, there is a strong and unique bidirectional relation between mobility and data communication in vehicular networks.

However, most traffic and network simulators still use the isolated communication approach, i.e., they do not exchange information in real time. Another technique uses a software for connecting traffic and network software to get interactions. Nevertheless, they suffer with latency requirements, due synchronization among models. This last method is computationally complex, because both simulators must be run simultaneously. Another issue appears because many network simulators do not support devices and the protocol stack proposed for VANETs.

Thereby, another approach is the formal verification using Probabilistic Model

Checking (PMC), a technique that automatically and exhaustively explores a model, verifying if it satisfies properties given in special types of logics. Preliminary studies show that model checking can be used to analyze Wireless Sensor Networks and VANETs. They consider important questions like non-determinism to broadcast the messages. However, traffic flow, computer networks and radio-propagation are necessary and rarely explored together in this field.

Aware of this gap, this work contributes to formal analysis in VANETs presenting a modeling structure with a microscopic granularity, which describes the traffic flow in details together with a stochastic way to represent the possibility of receiving a message with a probability p according to each vehicle's position. Furthermore, we proposing guidelines for building and verifying vehicular networks using our modeling. Thus, future works may use the suggested instructions to create the models with a similar abstraction level making the studies and results comparable.

To illustrate the use of the proposed model, applications and protocols for VANET have been used in this work. Thus, this thesis presents the necessary concepts as model checking, vehicular networks and virtual traffic light. The analytical models used in our guidelines are also introduced. Finally, some experimental models are implemented with the presented concepts.

Palavras-chave: Probabilistic model checking, VANET, Vehicular Network, Virtual Traffic Lights, VTL.

List of Figures

1.1	Vehicle-to-vehicle communications – Font [Herald Wheels, 2013]	2
1.2	Framework proposed	8
2.1	VANET architectures – Font [Alves and et al., 2009]	13
2.2	VANET projects – Font [Alves and et al., 2009]	14
2.3	WAVE frequency specifications – Font [Karagiannis et al., 2011]	15
2.4	Protocol stack suggested by IEEE 1609 – Font [Alves and et al., 2009]	16
2.5	V2V and V2I communication – Font [Hartenstein and Laberteaux, 2008]	19
3.1	VANET simulation environment - Adapted from [Boban and Vinhoza, 2011]	24
3.2	Simulation levels - Adapted by [Harri et al., 2009]	26
	(a) Origin-Destination matrix	26
	(b) Flow Model	26
	(c) Path modeling	26
3.3	Mobil notations - Font [Dirk Helbing and Martin, 2007]	30
3.4	Concept map for realistic mobility models – Font [Boban and Vinhoza, 2011]	33
3.5	Signal strength varying with distance and interference - Adapted by [Boulis et al., 2008]	35
3.6	Traffic lights communicate with adapt timing – Font [Gradinescu et al., 2007]	40
3.7	AVTL scheme – Font [Chou et al., 2013]	42
	(a) Proposed data flow	42
	(b) Leader selection	42
3.8	Crossroad with LOA and LONR lines depicted – Font [Santos et al., 2010]	43
3.9	Distributed VTL State Machine – Font [Santos et al., 2010]	44
3.10	VTL scheme – Font [Ferreira et al., 2010]	46
	(a) Algorithm VTL	46
	(b) VTL State Diagram	46
3.11	VTL – Font [Tonguz, 2011]	47

(a)	In-vehicle traffic lights concept	47
(b)	Average traffic flows using VTL	47
3.12	Layout scenario test – Font [Neudecker et al., 2012]	48
4.1	A Kripke structure and a computational path in it.	54
(a)	Generic Kripke structure with two states.	54
(b)	A computational path in the Kripke structure.	54
4.2	The symbolic representation of a transition.	55
4.3	A binary decision tree for the boolean function $(a \wedge b) \vee (c \wedge d)$	57
4.4	A reduced binary decision diagram.	59
4.5	Unfolding a Kripke structure in an infinite computational tree.	60
(a)	A Kripke structure.	60
(b)	An infinite computational tree.	60
4.6	The “Eventually” operator: $\mathbf{F} \phi$	61
4.7	The “Globally” operator: $\mathbf{G} \phi$	61
4.8	The “Next” operator: $\mathbf{X} \phi$	61
4.9	The “Until” operator: $\phi_1 \mathbf{U} \phi_2$	61
4.10	Different types of combinations of temporal operators.	62
(a)	$\mathbf{AF} \phi$ – for every path, ϕ is true in a future state.	62
(b)	$\mathbf{EF} \phi$ – there exists a path where ϕ is true in a future state.	62
(c)	$\mathbf{AG} \phi$ – for every path, ϕ is true in all future states.	62
(d)	$\mathbf{EG} \phi$ – there exists a path where ϕ is true in all future states.	62
4.11	An example of a CTMC model.	64
4.12	PRISM model of the vehicular movement.	66
4.13	An example of a state reward.	67
4.14	The adjacency matrix that explicitly represents the CTMC model.	72
4.15	A CTMC and its MTBDD representation	72
(a)	A CTMC model.	72
(b)	The MTBDD for the CTMC model.	72
4.16	A schematic of the four-way crossroad intersection with traffic lights which have been modeled using probabilistic model checking. Adapted from [Christian, 2009]	74
4.17	DEFINE commands for the communication channel modeling. One command represents that the channel is free and can be used, and the other one represents that a message collision has happened. Adapted from [Christian, 2009].	75
4.18	CaVi Architecture – Font [Fehnker et al., 2009]	76

4.19	Network example – Adapted by [Fehnker et al., 2009]	77
4.20	PRISM code for node 3	77
4.21	$rx_{i,j}$ constants	78
4.22	Examples of the SNR from node 3 to 1.	78
4.23	Examples of the $precv$ from node 2 to 3.	78
4.24	Probabilities “p” of connection.	78
4.25	Checking basic properties	79
4.26	Transition system for emergency message control. – Font [Konur and Fisher, 2011]	81
4.27	Checking basic properties – Adapted by [Konur and Fisher, 2011]	82
5.1	Mobility VANET Model	86
5.2	VTL Protocol PRISM Model	87
5.3	Semaphore, Lane and Timer Safety Properties	89
5.4	Semaphore and Lane Rewards and Quantitative Properties	89
5.5	Traffic Comparison of VTL and Timed Traffic Lights	90
5.6	Semaphore Interchange Property	91
5.7	Probability of Semaphore Interchange for each Protocol	92
5.8	Overtaking vehicle scenario	92
5.9	Variables of model	93
5.10	Initial states for the model	93
5.11	IDM model implementation	94
5.12	Modules implementation	95
5.13	Correctness properties	96
5.14	Counterexample to situations without accidents	97
5.15	Properties of overtake	97
5.16	Movement and Lane Rewards and Quantitative Properties	98
5.17	Scenario analysis	99
	(a) Movement in a normal overtake	99
	(b) Movement with accident	99
5.18	Acceleration and speed analysis in a free car-crash overtake	100
	(a) Acceleration evolution	100
	(b) Speed evolution	100
5.19	Time Rewards and Properties	100
5.20	Initial state for counterexample (unsuccessful overtaking)	101
	(a) Minimum time of collision	101
	(b) Maximum time of collision	101

5.21	Initial state for counterexample (successful overtaking)	101
(a)	Minimum time without collision	101
(b)	Maximum time without collision	101
6.1	Guideline steps	104
6.2	Proposed Architecture	112
6.3	Encoding of Proposed Architecture	118
6.4	Execution path of the proposed architecture	120
6.5	Acceleration reward	121
6.6	Rate reward	121
6.7	Probability reward	121
6.8	Speed reward	121
6.9	Position reward	121
6.10	Received msgs.	121
7.1	$c1$ overtakes the Leader , while $c2$ comes in the opposite direction.	128
7.2	Movement modules implementation in the PRISM language.	129
7.3	Network module implementation in the PRISM language.	129
7.4	Synchronize Module implementation in the PRISM language.	130
7.5	Acceleration reward	130
7.6	Power Transm. reward	130
7.7	Sent msg reward	130
7.8	Prob. reward	130
7.9	Distance reward	130
7.10	The average minimum and maximum of power transmission.	131
7.11	Signal propagation analysis of $c1$ and $c2$ communicating.	132
(a)	Probability of successful reception when only $c1$ sending.	132
(b)	Probability of successful reception when $c1$ and $c2$ sending.	132
(c)	Probability of successful reception in flooding protocol.	132
7.12	Probability of direct message reception	134
7.13	The acceleration behavior in two different scenarios.	136
(a)	Larger distance to stop.	136
(b)	Shorter distance to stop.	136

List of Tables

2.1	Nomenclature of WAVE architecture – Adapted [Alves and et al., 2009] . . .	16
2.2	Application requirements examples – Adapted [Karagiannis et al., 2011] . . .	20
3.1	Parameters of the IDModel. Adapted from Kesting et al. [2010]	29
3.2	Parameters of the Mobil Model. Font– [Dirk Helbing and Martin, 2007] . . .	32
3.3	Analysis of VTL for infrastructure and in-vehicle features.	50
3.4	Analysis of VTL for measure of effectiveness and tests.	50
4.1	Model Checking Research	83
5.1	Model statistics for different cars per lane (<i>timer</i> = 15).	88
5.2	Model statistics for different semaphore timers (<i>cars per lane</i> = 2).	88
5.3	Model statistics for different proprieties.	95
6.1	Proposal to final report structure	124
7.1	Property Results	133

Contents

Acknowledgments	ix
Abstract	xiii
List of Figures	xv
List of Tables	xix
1 Introduction	1
1.1 Motivation	1
1.1.1 Analysis of Computer Networks	3
1.2 Objectives and Intended Contributions	6
1.3 An Overview of the Proposed Approach	7
1.4 Outline of the Thesis	9
1.5 Publications	10
2 Vehicular Network	11
2.1 Introduction	11
2.2 Protocols and Standards	12
2.3 Special Characteristics of Vehicular Networks and Technical Challenges	16
2.3.1 Vehicular Network Potential Applications and Services	17
2.3.2 Simulation	21
2.4 Conclusions	21
3 VANET Simulation	23
3.1 Introduction	23
3.2 Mobility model	27
3.2.1 Lane-Changing Model - Model MOBIL	29
3.2.2 Framework for Realistic Vehicular Mobility Models	32

3.3	Data exchange (Network) model	33
3.4	Radio-propagation model	34
3.5	Discussion	37
3.6	A Survey on Virtual Traffic Lights	38
3.7	Virtual Traffic Lights (VTL)	39
3.7.1	Adaptive VTL Using Car-to-Car Communication	40
3.7.2	Adaptive VTL Based on VANETS for Mitigating Congestion in Smart City	41
3.7.3	Distributed VTL System	43
3.7.4	Self-Organized Traffic Control	45
3.8	Qualitative Evaluation of Safety in VTL Protocols	47
3.9	Conclusions	49
4	Probabilistic Model Checking	51
4.1	Introduction	51
4.2	Symbolic Model Checking	52
4.2.1	Kripke Structure	53
4.2.2	Kripke Structure Representation	54
4.2.3	Binary Decision Diagrams	56
4.2.4	Temporal Logic	59
4.3	Probabilistic Model Checking	62
4.3.1	Probabilistic Logics	64
4.3.2	PRISM	65
4.4	Applying Model Checking to Wireless Sensor Networks and Vehicular Networks	73
4.5	Reliable Model Checking for WSNs	74
4.6	Graphical modelling for simulation and formal analysis of wireless network protocols	76
4.6.1	Analytic model to PRISM	77
4.7	Formal Analysis of a VANET Congestion Control Protocol through Probabilistic Verification	79
4.8	Conclusion	82
5	Our Models to Verify Movements in VANETs	85
5.1	Macroscopic model	85
5.2	Results for the macroscopic model	87
5.2.1	Semaphore, Lane and Timer Safety Properties	88

5.2.2	Semaphore and Lane Quantitative Properties	89
5.2.3	Traffic Comparison of VTL and Timed Traffic Lights	90
5.2.4	The Influence of Traffic and Timer in Semaphore Interchange	91
5.3	Microscopic model	91
5.4	Results for the microscopic model	95
5.4.1	Car-crash situation	96
5.5	Conclusions	101
6	Our guidelines to analyze VANETs in model checking	103
6.1	Introduction	103
6.2	Planning	105
6.3	Design	107
6.3.1	Type of models	108
6.3.2	Level of detail	109
6.3.3	Initial conditional and limits	109
6.3.4	Measure units	110
6.3.5	Architecture	111
6.3.6	Model checker	112
6.4	Implementation and Analysis	114
6.4.1	Encoding (Guiding Principle I.4)	117
6.4.2	Basic analysis (Guiding Principle I.5)	120
6.5	Final report	123
6.6	Conclusion	125
7	Our Complete Model to Verify VANETs	127
7.1	Proposed Method	127
7.2	Results	131
7.3	Conclusion	136
8	Conclusions	139
8.1	Final Remarks	139
8.2	Our Contributions	141
8.3	Future work	142
	Bibliography	145

Chapter 1

Introduction

1.1 Motivation

Practically all major cities of the world are facing problems of traffic congestion. One cause is the rapid growth of the population, consequently increasing number of cars on roads. As a result, parking problems, long delays in getting to and from places and accidents are more and more constant. According to Sheng et al. [2011] relying only on the construction of transport infrastructure does not fundamentally solve the existing transportation problems. So many countries have been adopting Intelligent Traffic Systems (ITS).

These systems have as some of their main objectives to reduce the number of traffic accidents, the cost of transport and CO_2 emissions in the atmosphere [Ferreira et al., 2010]. These systems make intensive use of communication among vehicles, which is possible through the Vehicular Ad-hoc Networks (VANETs). This type of network represents a particular class of Mobile Ad-Hoc Networks (MANETs). VANETs are distributed self-organizing communication networks, mainly characterized by their high speed, which poses several challenges [Harri et al., 2009].

VANETs provide a wide variety of applications that can be divided in three classes [Vehicle Safety Communications Consortium, 2004]: (1) **Traffic Safety** — which has preventive and emergency objectives. The main challenge is to rapidly disseminate the information so that the driver has time to react to an unexpected situation; (2) **Entertainment** — these include adaptations of Internet applications for vehicular networks; and (3) **Driver Assistance** — related to information retrieval that helps the driver in searching or automating services, for example, Intersection assistance.

Intersection Management has been studied by several authors in order to optimize

traffic flow [Bengtsson et al., 1995; Lomuscio et al., 2010]. Ferreira et al. [2010] presents an approach that proposes replacing the fixed infrastructure for a control in which the vehicles manage the flow temporarily. This method was named Virtual Traffic Light (VTL).



Figure 1.1: Vehicle-to-vehicle communications – Font [Herald Wheels, 2013]

Figure 1.1 illustrates a VANET environment connecting several vehicles. It allows the exchange of information. Hence, vehicles approaching a blind intersection could warn each other of their existence. Thus, a virtual traffic light as mentioned above could be implemented. However, the protocol must be secure and ensure that no accidents will occur. Thus, it is necessary to have answers to questions like, what is the probability of accidents using a certain protocol? Which should be the speed and acceleration of the involved vehicles in order to avoid an accident? Those questions have to be answered taking into account all possible behaviors.

Current research in this area frequently analyzes VANETs using simulators to evaluate their behavior. However, simulation examines only a subset of possible behaviors, which can lead to incomplete analysis [Hartenstein et al., 2010]. Furthermore, works such as Alves and et al. [2009] and Boban and Vinhoza [2011] report that VANET simulators, despite constant evolution, have not reached an ideal point, because they need to integrate the mobility of the nodes, the communication protocols and the signals propagation. This is necessary since, the great benefit of VANET is to transmit information about the traffic in order to modify the routes of vehicles. However this benefit has become a challenge for simulation. Thus, two hitherto unconnected worlds must now work together, network and traffic simulators.

Currently, the vehicular traffic flow simulators can produce trace files that are given as input to a network simulator, or the traffic flow simulator must be coupled

with the network software to allow feedback from communication to vehicular traffic behavior. Another technique is to use a software to do the interface among established Network and Mobility simulator. In this context, there are challenges that must be addressed by the research community [Hartenstein et al., 2010; Hartenstein and Laberteaux, 2008; Boban and Vinhoza, 2011; Alves and et al., 2009]: (1) Specifications of APIs for coupling traffic flow and networking simulators (2) Modeling how drivers react to the additional information provided by VANETs (3) Benchmark definitions to make simulation studies and results comparable (4) Define the required level of accuracy in simulation according to application type. (5) Most simulators do not properly represent the hardware and protocols of vehicular networks (e.g. Wave protocol and modern chipsets) (6) Modeling and analyzing the effect of large scale fading, in particular of moving radio-wave obstacles like a truck between two cars, requires more attention from the communications community. (7) Real-world measurements show that deterministic radio propagation models should be avoided, because they do not capture the probabilistic effects of small scale fading that have a significant impact on packet reception.

1.1.1 Analysis of Computer Networks

Despite of simulation to be predominant in vehicular networks tests, others techniques like measurements and analytical/mathematical modeling can be used. According to Jain [1991], when the system under investigation already exists and it is necessary to improve a product, measurement techniques can be used. On the other hand, when the system does not exist or is too difficult to deal with, a model must be developed.

A mathematical model is an approximate representation of a physical situation. A useful model explains all relevant aspects of a given situation. When adequately constructed, models therefore allow the analyst to avoid the costs of experimentation [Leon-Garcia, 2008].

According to Puigjaner [2003], the main existing analytical techniques are based on the following formalisms: Queuing networks, Petri nets and Process algebras, with some variants like Stochastic Petri Nets and Stochastic Automata Networks. These techniques are basically special cases of Markov chain, which is a collection of random variables having the property that, given the present, the future is conditionally independent of the past [Kwiatkowska et al., 2004]. Here, it is needful to briefly introduce some of these methods [Obiniyi et al., 2014]:

Queuing Networks (QN). Queuing theory is the mathematical study of waiting lines. A model is constructed so that queue lengths and waiting times can be

predicted. It has found useful applications in telecommunications, traffic engineering, computing and the design of diverse building structures. A queuing system describes the system as a unique resource while a queuing network describes the system as a set of interacting resources [Yousefi et al., 2008].

Petri Nets (PN) are essentially a graphical and mathematical modeling tool applicable to many systems. Petri nets are directed graphs with two types of nodes, places and transitions, and unidirectional arcs between them. Generally, tokens move between places according to the firing rules imposed by the transitions [Puigjaner, 2003]. PN have been found to be very useful in the study of behavior of many real-world scenarios involving concurrency, sequencing, synchronization and conflict. Essentially in computer networks, PN can be used to describe and verify communication protocols [Puigjaner, 2003; Jahanian et al., 2015]. Other techniques are derived from PN, that is the case of Stochastic Petri Nets (SPN), which are Petri nets where exponentially distributed firing time is attached to each transition.

Stochastic Automata Network (SAN) is a formalism for the definition and the solution of complex systems with a very large state space. The main advantage of using SAN is its memory efficiency [Plateau and Stewart, 1997]. The basic idea is to represent a whole system by a collection of subsystems with an independent behavior and occasional inter-dependencies. Each subsystem is described as a stochastic automaton which represents a certain number of states, with rules or probability functions that govern the movements from one state of the automata to the other. SAN has been used efficiently in areas like machine decomposition, systolic computing, neural computing and IP networks [Othman et al., 2009; Plateau and Atif, 1997].

Performance Evaluation Process Algebra (PEPA) is a stochastic process algebra designed for modeling computer and communication systems. In PEPA the actions are assumed to have a duration or delay. Rates are drawn from the exponential distribution and PEPA models are finite-state and so give rise to a stochastic process. Hence it can be used to study quantitative properties of models of computer and communication systems. The main advantage of characterizing the corresponding class of PEPA models is that by “lifting” the definition from the stochastic process level in to a formally defined high-level modeling paradigm, automatic detection of these structures is facilitated when they occur [Hillston, 1996, 2005]. PEPA has been used in different areas including networks [Edwards, 2001; Fourneau et al., 2002], mobility model [Hillston and Ribaudó, 2004] and automotive systems [Argent-Katwala et al., 2008].

These aforementioned techniques are characterized by Obiniyi et al. [2014]; Baier et al. [2010] as methods of performance evaluation. These analysis are a branch

of computer science studies, which investigate the perceived performance of systems based on an architectural system description and a workload model [Baier et al., 2010]. According to Hermanns and Katoen [2001], performance evaluation aims at analyzing quantitative system aspects that are related to its performance and dependability. There are a number of reasons performance analysis study may be undertaken as stated in [Obiniyi et al., 2014]:

1. To find out performance bottlenecks in existing systems and develop improvements.
2. For capacity planning: for instance, how much resources should be spent to obtain some desired level of service quality?
3. For performance comparison of systems, algorithms and protocols; for example, given two protocols which one is better and in which respect?
4. To verify the claims of product designers and manufacturers or service providers; e.g. is the Internet service provider able to guarantee a certain minimum bandwidth as promised?
5. For predicting the performance at future workloads or operating conditions (forecasting).

Nevertheless, an important and complementary issue to performance is correctness [Baier et al., 2010]. Here, the central question is whether a system works conforming their requirements and does not contain any flaws, that is the type of questions which our work wants to answer.

Obiniyi et al. [2014] and Clarke et al. [1999] state that a prominent discipline in computer science to assure the absence of errors is model checking, a highly automated model-based technique to check whether a system model, which represents the possible system behavior, satisfies a property describing the desirable behavior. Typically, properties are expressed in temporal extensions of propositional logic, and system behavior is captured by Kripke structures, that is, finite-state automata with labeled states.

The strength of model checking which we have judged important to our work, is the ability to generate diagnostic feedback in the form of counterexamples (such as error traces) in case a property is refuted. This information is highly relevant to find flaws in the model and in the real system [Obiniyi et al., 2014].

Furthermore, the usage of temporal logics in model checking gives an important advantage to specify properties of interest at the same abstraction level as the modeling

with a high degree of expressiveness and flexibility [Obiniyi et al., 2014]. Nesting formulas yields a simple mechanism to specify complex measures in a succinct manner. A property like “the probability to reach a state within 25 seconds that almost surely stays safe for the next 10 seconds, via legal states only exceeds 0.5” boils down to $\mathcal{P}_{\geq 0.5}[\textit{“legal”} \mathbf{U}^{\leq 25} \mathcal{P}_{=1}[\mathbf{X}^{\leq 10} \textit{“safe”}]]$ and better, the meaning of the above formula is precise [Obiniyi et al., 2014; Clarke et al., 1999].

Yet according to [Obiniyi et al., 2014; Baier et al., 2010], perhaps the largest advantage of model checking to perform analysis is that all algorithmic details, all detailed and non-trivial numerical computation steps are hidden from the user. Without any expert knowledge on, say, numerical analysis techniques for CTMCs. This fact, together with the advantages mentioned above, as well as the act to be exhaustive, to support multiple types of models and reward structures (which could be used to do some performance analysis) became model checking an appropriate technique to exams correctness in VANETs.

There are some works in the literature about VANET and their protocols. However, in preliminary studies we pointed that few analysis treat the non-determinism of the message broadcast, the network protocol and the vehicles movement together. In other words, the focus of studied works was to verify the communication protocol. Nevertheless, it is important verifying networks considering not only communication by itself, but also implemented functionality. Thus, it is often necessary to model communication and other important system components to verify functionality [Christian, 2009].

1.2 Objectives and Intended Contributions

Probabilistic Model Checking (PMC) is a formal, exhaustive and automatic technique for modeling and analyzing stochastic systems [Clarke et al., 1999; Kwiatkowska et al., 2011]. PMC performs an automatic analysis of systems in which system properties are expressed in probabilistic logics, and are verified by the exhaustive enumeration of all reachable states. PMC may answer questions such as “What is the probability of the occurrence of a certain event?”. This is ideal for dynamic systems that exhibit stochastic phenomena such as VANETs. PMC verification is performed by (1) specifying the properties that the system must satisfy; (2) constructing the formal model of the system, which should capture all the essential properties and (3) running the model checker to validate the specified properties.

This work has the objective to model and analyze one specific protocol or

applications using more than one to VANETs. The idea is to provide a guideline for modeling which includes vehicular movement, non-deterministic message broadcast and network protocols. Thus, we will consider not only communication, but all the functionality of an application. For instance, to virtual traffic light algorithm, a leader election protocol is crucial, however, the protocol evaluation is based on optimistic and ideal assumptions, i.e., the studies neglect the existence of radio obstacles, and consider a perfect communications system, namely, every broadcast message is received by every vehicle. The analysis made by most researches focus in the evaluation of large scale traffic efficiency. It is unusual to investigate the reliability of these new VTLs through exhaustive methods or formal proofs. The researches also have difficulty to execute tests because simulation does not support implementation details for this type of application. Thus, almost all works had to build their own simulators or developed an interface able to do the integration between motion and network.

1.3 An Overview of the Proposed Approach

The goal is to provide examples and guidelines to model and analyze vehicular networks in a complete way using model checking. Figure 1.2 shows an abstraction of the proposed idea. This model is divided in three wide groups (gear) implemented by modules and/or formulas in some model checker language. The groups exchange data with each other and may change their behavior according to interaction. We have been moving towards groups with higher cohesion and looser coupling to represent motion, signal-propagation and network. Thus, we can change parts of the modeling according to our need. For instance, we may change modules responsible for a urban street to a highway, or changing the code responsible for signal propagation from expressions for the log-normal shadowing path loss to Nakagami model ([Van Eenennaam, 2008]).

We have proposed a microscopic model, which describes the traffic flow in details. With regard to the implementation, the Cars Following Models (CFMs) are probably the most popular class of driver model. CFMs usually represent time, position, speed, and even acceleration as continuous functions, but most have been extended to provide discrete formulations. One of most prolific CFM is Intelligent driver model (IDM) [Hartenstein et al., 2010].

The IDM shows a crash-free collective dynamics, exhibits controllable stability properties, and implements an intelligent braking strategy with smooth transitions between acceleration and deceleration behavior. The IDM acceleration is a continuous function incorporating different driving modes for all velocities in freeway traffic as well

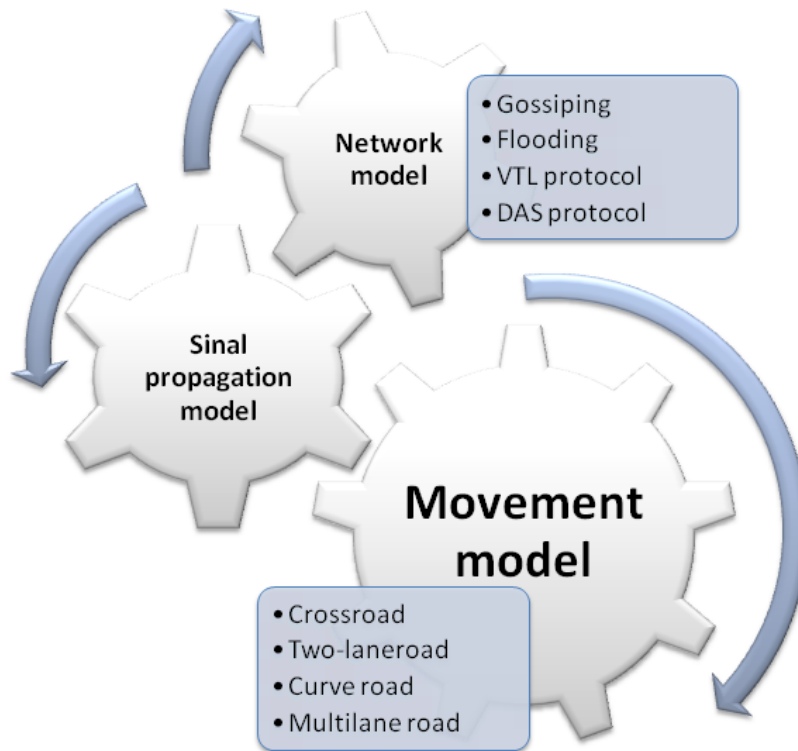


Figure 1.2: Framework proposed

as city traffic. Besides the (bumper-to-bumper) distance s to the leading vehicle and the actual speed v , the IDM also takes into account the velocity difference (approaching rate) to the leading vehicle [Kesting et al., 2010].

In order to adequately model VANETs and take into account the unique characteristics of environment, we propose the mobility modeling along with signal propagation using a stochastic model that represents the possibility to receive a message with a probability p . The link probability is calculated within arbitrary networks to take into account the distance between nodes and their relative clustering. The behavior of the wireless links is based on an empirically validated model proposed by Zuniga and Krishnamachari [2004]. The difference in behavior whether one node or several nodes broadcast at the same time is accounted for in the link probabilities. The method computes the various quantities such as the transmission powers, the signal-to-noise ratios and thresholds for each node, taking into account their actual pairwise separations.

1.4 Outline of the Thesis

This thesis is structured in the following chapters:

- Chapter 2 provides basic concepts on Vehicular networks, a novel and adaptive Ad hoc Networks. These networks are created among vehicles or between vehicles and fixed infrastructure located on the streets or roads edges. Thus, protocols, architectures and standards are presented. Special characteristics and technical challenges also are reported. Finally, the potential applications and services are pointed.
- Chapter 3 describes basic concepts of vehicular networks simulation, which minimizes costs and reduces time of the experiments. However it shows weaknesses. Thus, challenges and adopted solutions are presented. A microscopic mobility model and a signal propagation model used by simulators are presented. In the remainder of the chapter, we show how the analysis through simulation have been conducted. As an example, we choose the studies about Virtual Traffic Light, since their use are directly involved with human lives.
- Chapter 4 presents model checking, a formal verification approach which allows verifying if a model satisfies a set of logic properties. Symbolic model checking is first described, covering its symbolic representation and temporal logics. Since our models are stochastic, probabilistic model checking is covered, including its probabilistic and reward-based logics. Model checkers of Markov chains are reviewed. The PRISM model checking tool was used in this work, therefore we have included a brief description of its features. To finish, we present the related work of this project. This part presents three works using model checking in Wireless Sensor Networks and VANET. They show how the problem has been modeled.
- Chapter 5 presents two case studies proposed in this work to model movement using the PRISM model checker. The first one is a macroscopic model of an virtual traffic light managing a crossroad. It shows the feasibility of formal methods to analyze vehicular networks. The other model covers a microscopic representation which will be used in our guidelines to represent motions.
- Chapter 6 describes the guidelines proposed by our work. In this point, the steps from planning to the final report are presented. In addition, an example of source code in PRISM language is shown.

- Chapter 7 exemplifies our proposed architecture to modeling VANETs. We have applied a modeling structure presented in our guidelines which includes mobility, communication and signal propagation modules. We present an analysis of a vehicular warning system involving some automobiles.
- Chapter 8 covers final remarks and conclusions of our research. Furthermore, it exposes future work.

1.5 Publications

This thesis has generated the following publication and therefore contains material from it:

- [Ferreira et al., 2012b]: Ferreira, B. and Braz, F. A. F. and Campos, S. V. A. (2012). A probabilistic model checking approach to investigate vehicular networks. In Proc. 15th Brazilian Symposium Formal Methods;
- [Ferreira et al., 2014b]: Ferreira, B. and Braz, F. A. F. and Campos, S. V. A. (2014). A Probabilistic Model Checking Analysis of a Realistic Vehicular Networks Mobility Model. In Proc. 17th Brazilian Symposium Formal Methods;
- [Ferreira et al., 2015a]: Ferreira, B. and Braz, F. A. F. and Loureiro, A. A. F. and Campos, S. V. A. (2015). A Probabilistic Model Checking Analysis of Vehicular Ad-Hoc Networks. In Proc. IEEE 81st Vehicular Technology Conference;
- [Ferreira et al., 2015b]: Ferreira, B. and Cunha, F. and Mini, R. and Braz, F. A. F., and Loureiro, A. A. F. and Campos, S. V. A. (2015). Intelligent Service to Perform Overtaking in Vehicular Networks. In Proc. The 20th IEEE Symposium on Computers and Communications - ISCC2015;

Chapter 2

Vehicular Network

Outline. In this chapter we present some of the background on Vehicular Network that is relevant to this thesis. We introduce protocols, architectures and standards adopted in VANET, which are an important source to understand how we will model the network of the architecture shown in Figure 1.2. We also describe characteristics, technical challenges and some potential applications and services. Finally, we consider some simulation outlook.

2.1 Introduction

Automobiles have been incorporating several technological advances such as increasingly sophisticated sensors and actuators. A processor uses sensor data to control different systems on a vehicle through the use of actuators, which are electromechanical devices. This actuators can for example, adjust engine idle speed, change suspension height or regulate the fuel metered into the engine [Toyota, 2012]. This control provides the emergence of technologies such a smart breaking systems, driver fatigue detection and cruise control, the goal is to improve the experience of the driver and passengers.

In addition, the next technological evolution is the use of communication architectures to enable interaction between different vehicles. Such architectures and applications form an Intelligent Transportation System (ITS), which works in an environment formed by users in traffic [Alves and et al., 2009].

The vehicular communication used in ITS is called Vehicular Networks. These networks are created among vehicles or between vehicles and a fixed infrastructure located on the streets or roads edges. However, vehicular networks have a number of challenges to their widespread adoption. Among them are the high node mobility,

the dynamic scenarios, and scalability in number of nodes. The fading of connectivity during data transmission and reduced time in which two nodes remain in contact are other challenges. In this scenario, protocols designed for other wireless networks, such as Mobile Ad hoc (MANET), are not suitable [Alves and et al., 2009].

Vehicular ad hoc networks (VANETs) are emerging as a new class of wireless networks, spontaneously formed among moving vehicles equipped with wireless interfaces that could have similar or different radio interface technologies from short-range to medium-range communication systems. According to Hassnaa Moustafa [2009], VANET is a form of mobile ad hoc network that provides communications among vehicles following a specific architecture, which defines the way how the nodes organize and communicate with each other. Three main architectures are [Alves and et al., 2009; Hassnaa Moustafa, 2009]:

1. **Infra-structured network:** it employs static nodes distributed along streets or roads. These static nodes serve as access points in IEEE 802.11 networks. The advantage is the increased connectivity and the possibility of communication with other networks such as the Internet. The connectivity, however, is only guaranteed by a large number of fixed elements, which can raise the cost of the network. The wired backbone mode also is called V2I (Vehicle-to-Infrastructure)
2. **Wireless network:** vehicles communicate with no external support or centralizing element. Therefore, vehicles act as routers and forward messages through multiple hops. This is the simplest configuration, since it does not require any infrastructure. It has the network connectivity as major drawback, because it depends on the density and mobility pattern of vehicles. This model is known as V2V (Vehicle-to-Vehicle).
3. **Hybrid Architecture:** architecture that does not rely on a fixed infrastructure in a constant manner, but can exploit it for improved performance and service access when it is available. In this latter case, vehicles can communicate with the infrastructure either in a single hop or multihop fashion according to the vehicles' positions with respect to the point of attachment with the infrastructure. Figure 2.1 illustrates the three methods.

2.2 Protocols and Standards

Reducing the number of accidents may in turn reduce the number of traffic jams, which could reduce the level of environmental impact and improve the welfare of the



Figure 2.1: VANET architectures – Font [Alves and et al., 2009]

population. With these goals, various projects are underway or recently completed and several consortia were set up to explore the potential of VANETs [Hartenstein and Laberteaux, 2008]. These consortia projects involve several areas, including the automotive industry, the road operators, tolling agencies and national governments. A set of milestones presented in Figure 2.2 shows the evolution of the VANET research around the globe. These include the consortia like Vehicle Safety Consortium [VSC-US, 2011], Collision Avoidance Metrics Partnership [CAMP-US, 2013], Car-to-Car Communication Consortium [C2C-CC (EU), 2013], Advanced Safety Vehicle-ASV Program [NASVA-JP, 2013] and Large-scale Vehicle Infrastructure Integration Consortium [VII-US, 2012].

In this context, the Dedicated Short-range Communications System (DSRC) has emerged in North America. The Car-to-Car Communication Consortium (C2C-CC) has been initiated in Europe by car manufacturers. In Japan, five related government bodies jointly finalized a “System Architecture for ITS”, since, several projects have been developed, including the ASV Programme (Advanced Safety Vehicle) [Furukawa, 2005; Paromtchik and Laugier, 2007] which aims to develop methods and devices to improve the safety of transportation system. Despite having the same objective, different countries have adopted different specification and technology. An example, is the distinct frequency spectrum defined by projects as depicted the Figure 2.3.

The communication standard VANET was created by IEEE and was called WAVE (Wireless Access in the Vehicular Environment). WAVE is defined by the IEEE 1609 family, and the specification IEEE 802.11p. The latter defines the physical layer and medium access control (MAC). It is based on IEEE 802.11a standard created for local networks (LAN), which operates in a frequency band similar to the vehicular networks. As shown in Figure 2.4, the IEEE 1609 family is not restricted to the lower layers. The

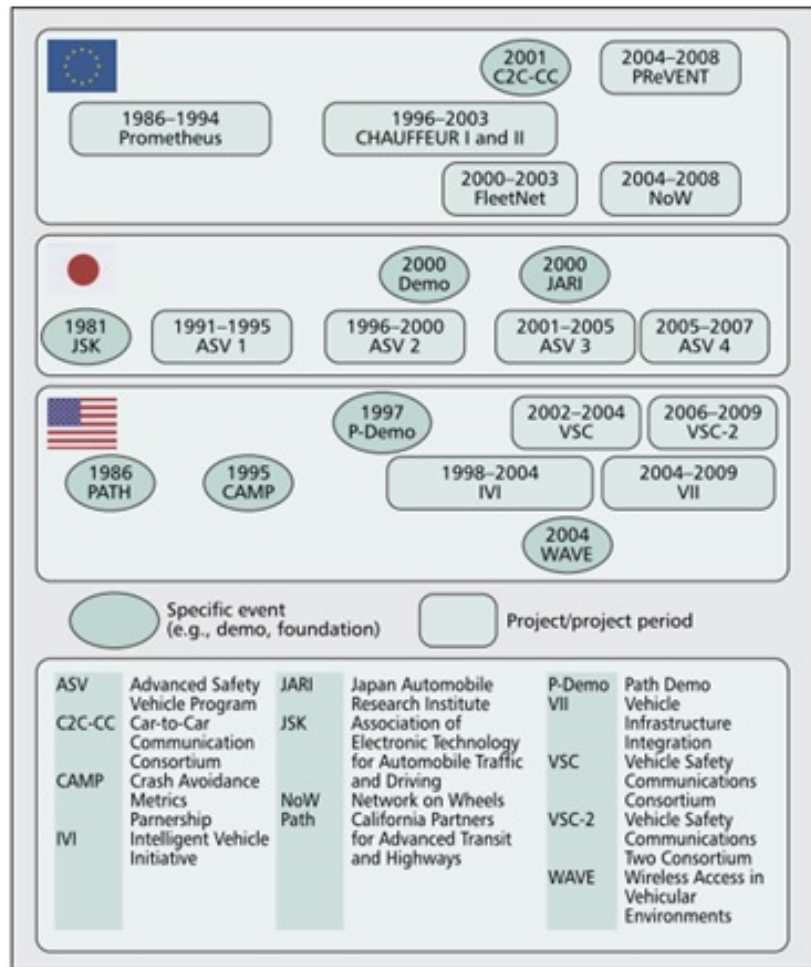


Figure 2.2: VANET projects – Font [Alves and et al., 2009]

Institute of Electrical and Electronics Engineers defined other points of the protocol stack including a network layer to be an alternative to the IP protocol, safety features for application DSRC operation and multiple communication channels [Alves and et al., 2009]. The protocol stack as suggested by the IEEE 1609 contains [Karagiannis et al., 2011]:

- **IEEE 1609.1:** describes an application that allows the interaction of an On Board Unit (OBU) with limited computing resources and complex processing running outside the OBU, in order to give the impression that the application is running on the OBU;
- **IEEE 1609.2:** specifies the WAVE security concepts and defines secure message formats and their processing in addition to the circumstances for using secure message exchanges;

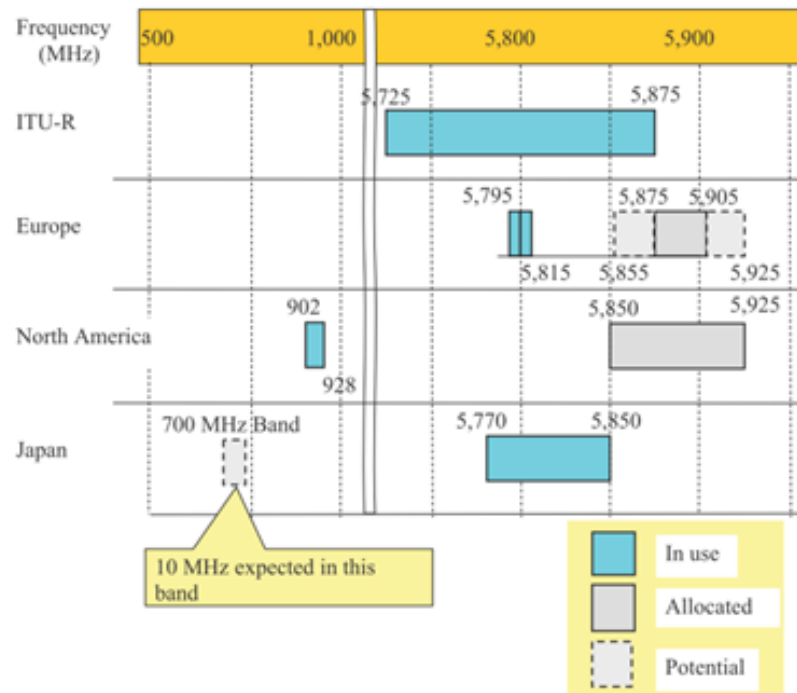


Figure 2.3: WAVE frequency specifications – Font [Karagiannis et al., 2011]

- **IEEE 1609.3:** provides routing and addressing services required at the WAVE network layer. WSMP (WAVE Short Message Protocol) provides routing and group addressing (via the WAVE Basic Service Set (WBSS)) to traffic safety and efficiency applications. It is used on both control and service channels. The communication type supported by WSMP is broadcast;
- **IEEE 1609.4:** provides multi-channel operation that has to be added to IEEE 802.11p.

The main objective of IEEE 1609 is to provide a standardized set of interfaces. Thus different automotive manufacturers can provide communications between the architectures shown in Figure 2.1. Furthermore, the standard should consider high mobility of nodes, i.e., communications must be completed in short time intervals [Alves and et al., 2009]. Therefore, IEEE 802.11p is based on an orthogonal frequency-division multiplexing (OFDM) in PHY layer, but uses 10 MHz channels as opposed to the 20 MHz channels for IEEE 802.11a. As a result, data rates range from 3 to 27 Mb/s for each channel, where lower rates are often preferred in order to obtain robust communication. Because the basic type of communication in a VANET is based on one-hop broadcasts, the IEEE 802.11 MAC can be understood as simple CSMA scheme. However, many parameters can influence the probability of packet reception. A partial list includes vehicular traffic density, radio channel conditions, data rate, transmission

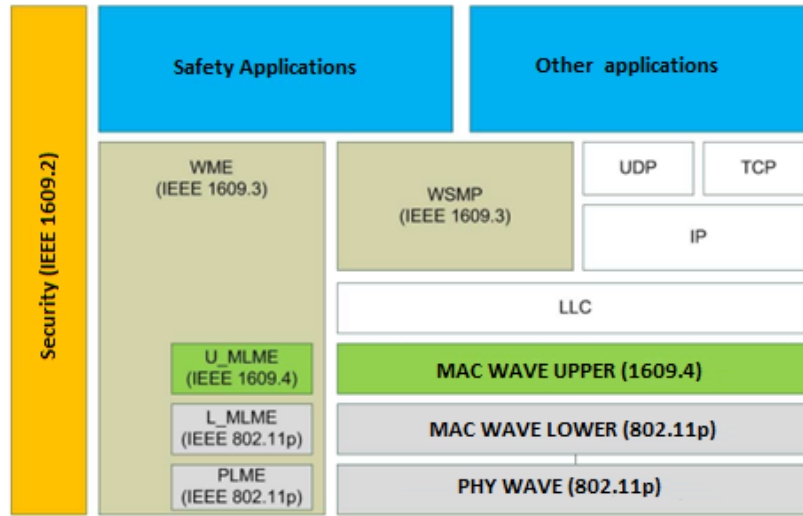


Figure 2.4: Protocol stack suggested by IEEE 1609 – Font [Alves and et al., 2009]

power, contention window sizes, and the prioritization of packets [Hartenstein and Laberteaux, 2008]. Finally, Table 2.1 describes the nomenclature used in IEEE 1609 family of standards.

Table 2.1: Nomenclature of WAVE architecture – Adapted [Alves and et al., 2009]

WAVE devices names	Description
On board Unit (OBU)	WAVE mobile device able to exchange information with other OBUs and RSUs
Road Side Unit (RSU)	WAVE static device able to exchange information with other OBUs and RSUs
WAVE Basic Service Set (WBSS)	Set of WAVE stations composed of a WBSS provider and zero or more WBSS users
WAVE Short Message (WSM)	Message broadcasted by the WSMS protocol
WBSS Provider	WBSS device able to broadcast WSMs
WBSS user	WSM receiver

2.3 Special Characteristics of Vehicular Networks and Technical Challenges

Vehicular networks have special behavior and features, which distinguish them from other types of mobile networks. VANET comes with unique appeals, as i) Unlimited transmission power, since each node (vehicle) can provide continuous power to computing and communication devices; ii) High computational capability: Indeed, vehicles can afford significant computing, communication, and sensing capabilities;

iii) **Predictable mobility:** Unlike classic mobile ad hoc networks, where it is hard to predict the nodes' mobility, vehicles tend to have very predictable movements since are (usually) limited to roadways. Positioning systems, like GPS, give the average speed, current speed, and road trajectory. Thus, the future position of a vehicle can be predicted [Nekovee, 2005].

However, vehicular networks have to cope with some challenging characteristics. A central challenge of VANETs is that applications should run reliably using decentralized communications. Many applications will be broadcasting information of interest to many surrounding cars. However, at least one shared control channel is required. This one-channel paradigm, together with the requirement for distributed control, leads to some of the key challenges of VANET design. The very well-known problem of hidden and exposed terminals is problematic. Clearly, medium access control (MAC) is a key issue in the design of VANETs [Hartenstein and Laberteaux, 2008]. Other challenges are [Blum et al., 2004]:

- **Potentially large scale:** Vehicular networks can in principle extend over the entire road network and so include many participants.
- **High mobility:** The environment is extremely dynamic, and includes extreme configurations: on highways, relative speeds of up to 200 km/h on low traffic freeways. On the other hand, in the city, relative speeds can reach up to 60 km/h and nodes' density can be very high, especially during rush hour.
- **Partitioned network:** Vehicular networks will be frequently partitioned. The dynamic nature of traffic may result in large inter-vehicle gaps in sparsely populated scenarios, and hence in several isolated clusters of nodes.
- **Network topology and connectivity:** VANET topology changes frequently as the links between nodes connect and disconnect very often. Indeed, the degree to which the network is connected is highly dependent on two factors: the range of wireless links and the fraction of participant vehicles, where only a fraction of vehicles on the road could be equipped with wireless interfaces.

2.3.1 Vehicular Network Potential Applications and Services

The primary vision of vehicular networks includes real-time and safety applications for drivers and passengers. It provides safety and gives essential tools to decide the best path along the way. Regarding application's potential, vehicular networks open new business opportunities for car manufacturers, network operators, service providers,

in terms of infrastructure deployment and commercialization [Hassnaa Moustafa, 2009]. VANET provides a wide variety of applications that can be divided in three classes [VSC-US, 2011; Karagiannis et al., 2011]:

1. **Traffic Safety Applications** – which have preventive and emergency objectives. The main challenge is to rapidly disseminate the information so that the driver has time to react to an unexpected situation. Two examples of active road safety applications are: i) *Intersection collision warning*: in this case study, the risk of lateral collisions for vehicles that are approaching road intersections is detected by vehicles or road side units. This information is signaled to the approaching vehicles in order to lessen the risk of lateral collisions. ii) *Lane change assistance*: the risk of lateral collisions for vehicles that are accomplishing a lane change with blind spot for trucks is reduced.
2. **Driver Assistance Applications** – related to information retrieval that helps the driver in searches or automated services, for example [ETSI, 2011]; i) *Speed management*: aims at assisting the driver to manage the vehicle speed for smooth driving and avoiding unnecessary stopping and ii) *Co-operative navigation*: this application manages the navigation of vehicles through cooperation among them. Some examples of this type are traffic information and recommended itinerary provisioning, co-operative adaptive cruise control.
3. **Entertainment Applications** – these include adaptations of Internet applications for vehicular networks. A typical application is Co-operative local services which focus on infotainment that can be obtained from locally based services such as point of interest notification, local electronic commerce and media downloading [C2C-CC, 2007; ETSI, 2011; Prj, 2008];

These three classes of VANET applications are not completely orthogonal: for example, reducing the number of accidents can in turn reduce the number of traffic jams, which could reduce the level of environmental impact (Figure 2.5). By vehicle-to-vehicle and vehicle-to-roadside communication, accidents can be avoided (e.g., by not colliding with a traffic jam) and traffic efficiency can be increased (e.g., by taking alternative routes) [Hartenstein and Laberteaux, 2008]. Thus, an application can serve to offer Safety and driver assistance.

ETSI [2011] identified performance requirements for vehicular networking applications. The requirements can be grouped into the following items:

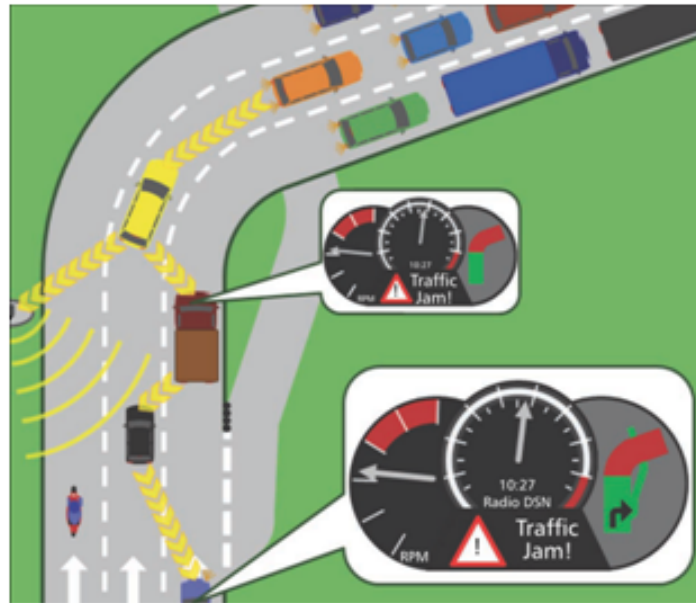


Figure 2.5: V2V and V2I communication – Font [Hartenstein and Laberteaux, 2008]

- a) **Strategic requirements:** These requirements are related to: (1) the level of vehicular network deployment, e.g., minimum penetration threshold and (2) strategies defined by governments and commissions.
- b) **Economical requirements:** These requirements are related to economical factors, such as business value once the minimum penetration value is reached, perceived customer value of the case study, purchase cost and ongoing cost and time needed for the global return of the invested financial resources.
- c) **System capabilities requirements:** These requirements are related to the system capabilities, which are: Radio communication capabilities, such as single hop radio communication range, used radio frequency channels, Network communication capabilities, such as mode of dissemination, congestion control, message priority. Vehicle absolute positioning capabilities, such as Global Navigation Satellite System (GNSS), e.g., Global Positioning System (GPS), beyond, others vehicle capabilities and vehicle communication security capabilities.
- d) **System performance requirements:** which are: related to the system performance, vehicle communication performance, vehicle positioning accuracy.
- e) **Organizational requirements:** associated with deployment, which are: IPv6 or IPv4 address allocation schemes, suitable organization to ensure interoperability between different Intelligent Transport Systems.

- f) **Legal requirements:** related to legal responsibilities, which are: support and respect of customer's privacy, support the responsibility of actors, support the lawful interception.
- g) **Standardization and certification requirements:** related to standardization and certification, which are: support of system standardization, support of Intelligent Transport System station standardization.

System performance requirements depend upon the class of application. For safety applications, the coverage distance varies from 300 meters to 2000 meters depending on the study case. But, the coverage distance associated with entertainment applications varies from 0 meters to full communication range [C2C-CC, 2007; ETSI, 2011]. Examples for some applications are shown in Table 2.2.

Table 2.2: Application requirements examples – Adapted [Karagiannis et al., 2011]

Applications classes	Study case	Communication mode	Minimum trans. freq.	Critical latency
Safety application	Intersection collision warning	Periodic msg. broadcasting	10 Hz	< 100 ms
	Lane change assistance	Co-operation awareness between cars	10 Hz	< 100 ms
Driver assistance	Green light optimal advisory	Periodic, permanent broadcasting msg.	10 Hz	< 100 ms
	Electronic toll collection	Internet vehicle and unicast full duplex session	1 Hz	< 200 ms
Entertainment	Vehicle software provisioning and update	Access to internet	1 Hz	< 500 ms

Intersection collision warning should follow these requisites. This service has been studied by several authors in order to optimize traffic flow [Santos et al., 2010; Gradinescu et al., 2007]. Ferreira et al. [2010] present an approach that proposes replacing the fixed infrastructure for a control in which vehicles manage the flow temporarily. This method was named Virtual Traffic Light (VTL). Some VTL concepts and development examples about VTL are presented in Section 3.6.

2.3.2 Simulation

Tests in vehicular networks require a large number of people, high costs and favorable weather conditions. Moreover, the repetition of a given experiment in an environment with many variables is hard. The use of simulation is an attractive alternative. It provides a controlled environment and spends less resources. But the reproduction of similar conditions which are found in real tests is a challenge. The simulation should represent the signals propagation and network protocols. Furthermore, it should deal with the mobility of nodes. Thus, specific mobility models must be developed [Alves and et al., 2009].

Most of the proposed solutions use a known network simulator in conjunction with a mobility simulator. This maybe involves other software to do the interface among them. An example is the TraNS (Traffic and Network Simulation Environment) [Piorkowski et al., 2008]. It is a simulation tool that uses the NS-2 and SUMO. This is possible due to TraCI (Traffic Control Interface), an interface module that connects the two simulators.

Specific simulators for vehicular networks have been proposed. Wang and Lin developed the simulator NCTUns [Wang and Lin, 2008], that has simulation modules for the IEEE 802.11p protocol. This simulator has an interface which allows the creation of nodes in the network and specific paths [Alves and et al., 2009]. Despite advances, this software and projects still have great challenges. Thus, developers have been improving the tools constantly. More concepts and features of simulation are presented in Chapter 3.

2.4 Conclusions

Vehicular network is a technology that will support several applications varying from global Internet services to road safety applications. This chapter introduced and discussed some possible applications.

Although many standard organizations are involved in the study and standardization of VANETs, these networks are considered as a technology under development that merits a lot of research. Moreover, a number of technical challenges need to be resolved for wide-scale deployment of these networks in the near future. Furthermore, this chapter showed important features of communication, which the network model presented in the architecture of the Figure 1.2 should have, for instance, aspects like latency and protocols to broadcast the messages.

Chapter 3

VANET Simulation

Outline. In this chapter we present some of the background on VANET Simulation that is relevant to this thesis. We introduce the evolution, development and features of the simulators. We also describe the Network, Radio-propagation and Mobility model including a Lane-Changing representation. A proposed framework for realistic vehicular mobility models also is introduced. Finally, the use of simulation examining different studies about Virtual Traffic light are presented to illustrate the problems of this analysis type.

3.1 Introduction

In order to validate the effectiveness of Intelligent Traffic Systems, it is necessary to evaluate their performance and communication protocols in real test environments. However, there are logistic difficulties, economic questions and technological limitations which make simulations a good choice for testing and validation of these protocols. The fields of computer networks and traffic engineering make extensive use of simulators. There are long established software such as NS-2 (The Network Simulator)¹ and SUMO (Simulation of Urban Mobility)². Since the introduction of vehicular networks, the integration of these two fields has recently become necessary [Hartenstein et al., 2010].

This integration is required due to inherent features of the strong coupling between communication and mobility in VANETs. Its use alter mobility patterns, which in turn change the data communication between vehicles. This ad hoc infrastructure compromises the correct message reception, which in turn changes its mobility. Therefore, there is a strong and unique bidirectional relation between mobility

¹NS-2. <http://www.isi.edu/nsnam/ns/>. Access date: October 17, 2016

²SUMO. <http://sumo.sourceforge.net/>. Access date: October 17, 2016

and data communication in VANETs. Thus, three distinct aspects must work together in order to achieve realistic tests [Boban and Vinhoza, 2011]: (1) **Mobility Models** represent vehicle movement, including mobility patterns and the interaction between vehicles (e.g. crossroad control); (2) **Network Models** describe the data exchange between vehicles, including MAC, routing and superior protocol layers; (3) **Signal Propagation Models** intend to reproduce the environment modeling involving fixed and mobile obstacles during the communication.

Figure 3.1 shows the relationship between these three models and techniques which could be used for the implementation of each model. These models can be applied in different forms by choosing the available techniques accordingly to the desired realism level. For further details on these mobility and signal propagation techniques, refer to [Harri et al., 2009] and [Khosroshahy, 2007], respectively.

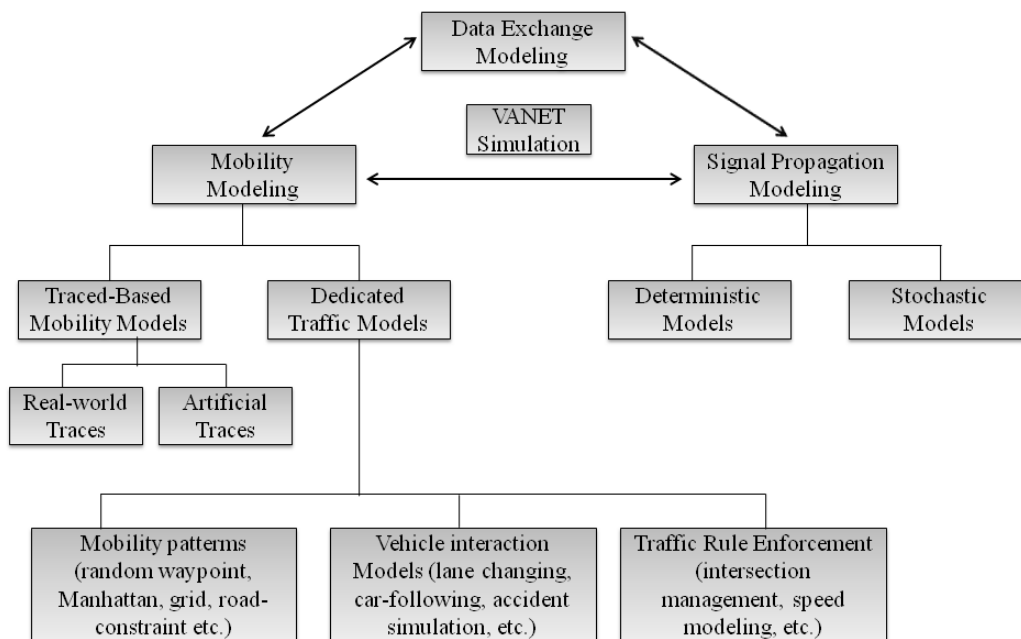


Figure 3.1: VANET simulation environment - Adapted from [Boban and Vinhoza, 2011]

As an example of realism level, an application directed to the traffic efficiency should perform its simulations in large scale, with less precision on the movement modeling. Network simulators should consider a full bidirectional interaction, increasing its granularity to an arbitrary packet loss rate and a data reception probability is enough.

On the other hand, traffic security applications require a considerable precision level of vehicular movement modeling, a high degree of realism in the network and

signal propagation models, although represented in a small scale with a simplified traffic model. A common point between the simulation of these types of applications is the communication between network and traffic simulators. These simulators have usually been developed in different periods and they are connected in distinct forms as described below [Harri et al., 2009]:

- **Isolated:** this was an initial approach in the communication between different models. The simulation traces for vehicular mobility are statically generated and then used by a network simulator. The main disadvantage is the difficulty to define real time interactions between the network and the traffic itself. However, its simplicity and universality are useful in some current applications.
- **Embedded:** in this approach, either a traffic vehicular simulator is inserted in a network simulator, or the opposite. This allows the duplex interaction between both simulators. However, this approach still is not ideal, because simplified network simulators are inserted in the mobility ones (for example, Bononi et al. [2006]), or detail-rich network simulators receive reduced vehicular traffic modules, such as Wang and Chou [2008].
- **Federated:** a traffic simulator is integrated with a network simulator through a communication interface, which controls the data exchange between them. Renowned and detail-rich simulators can be used, such as in Wu et al. [2005], which couples the traffic simulator CORSIM with the network simulator QualNet. Another example is presented in VGrid [2009], which integrates the traffic simulator SUMO with other network simulators such as NS-2 or OMNeT+++. The drawback is the communication latency between different simulators and the amount of parameters to configure the connection between the softwares.

The Isolated approach uses "Trace-based Models" (Figure 3.1) for mobility patterns, which can use artificial traces, with random movement on a plane or with real-world traces. This last one brings a certain level of realism, however this approach does not provide a connection feedback to the network model in order to change the vehicle movement. Ferreira et al. [2009] and Ho et al. [2007] use real-world traces and they are examples of this technique. The other two approaches, Embedded and Federated, use the "Dedicated Traffic Models", because they provide an improved ability to connect mobility and network models [Boban and Vinhoza, 2011]. Figure 3.1 shows several techniques used to implement this dedicated model. Thus, in order to guide the choices about what to use during simulation, Harri et al. [2009] proposes dividing the modeling

into three levels to facilitate the search for realism and efficiency requirements in ITS testing:

1. **Trip Modeling:** vehicular movement is defined by trips between Points of Interest (PoI) and can be done, for example, using the OD-Matrix (Origin-Destination), which selects PoIs in the traffic environment and builds a transition matrix to represent the correlations between routes. Figure 3.2a illustrates an OD-Matrix, where the transitions are probabilities to move from one vertex (PoI).
2. **Path Modeling:** once origin and destination have been defined, the route planner defines the instruction sequence of each vehicle in order to reach its destination. Path Modeling pre-computes or dynamically recomputes the preferred intersection sequence based on the optimization function (e.g. shortest or fastest path). Figure 3.2c distinguishes paths between PoIs for black and gray vehicles.

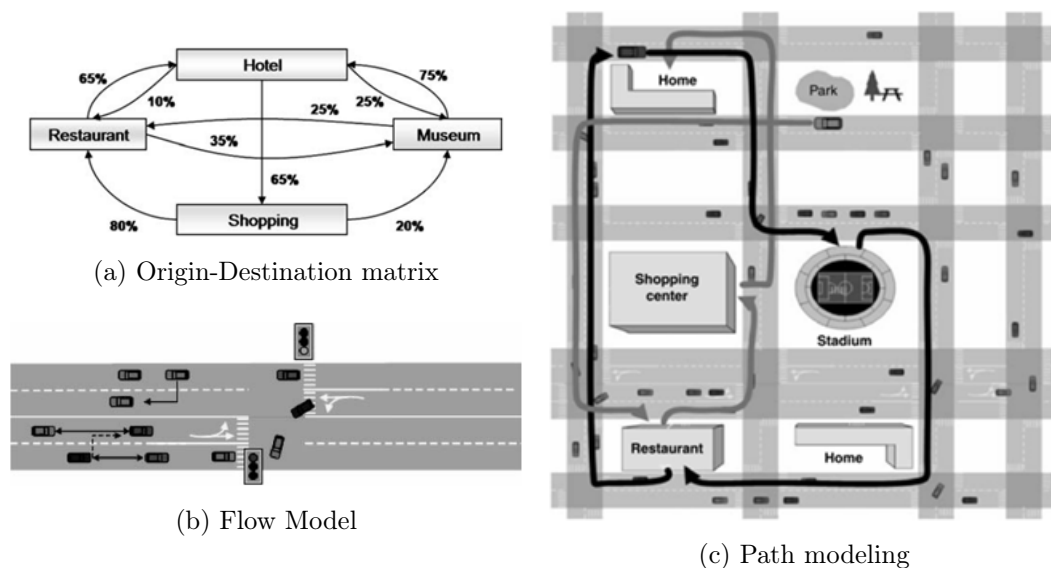


Figure 3.2: Simulation levels - Adapted by [Harri et al., 2009]

3. **Flow Modeling:** a detailed description of vehicular mobility is used to model the interaction of vehicles as a vehicle flow. Figure 3.2b illustrates the precision of these interactions for an intersection. Productive methods for the implementation of mobility models include [Helbing, 2001]: cellular automata [Nagel and Schreckenberg, 1992], intelligent driver model [Kesting et al., 2010] and gas-kinetic [Hoogendoorn and Bovy, 2001].

Another important aspect of simulations is the classification of the granularity for the mobility model, which can be: microscopic, mesoscopic and macroscopic. This aspects are detailed below [Harri et al., 2009]:

- **Microscopic:** traffic flow is described in detail, regarding the mobility parameters of specific vehicles and their interactions with other vehicles. These parameters usually are vehicle acceleration/deceleration, adjusted in order to maintain safe distances between vehicles. However, it is computationally expensive.
- **Macroscopic:** this category does not consider a specific vehicle for the mobility parameters – instead, it describes the class of vehicles as a flow or density. Inspired by the flow theory, it has the advantage of a reduced computational complexity compared to the microscopic models.
- **Mesoscopic:** this description uses traffic flow and an intermediate level of detail. Individual vehicles can be modeled, although using a macroscopic perspective. The objective is to benefit from the macroscopic scalability, however still providing details comparable to microscopic models.

3.2 Mobility model

At beginning, VANET mobility models were characterized by their simplicity and ease of implementation. However, simplified mobility models such Manhattan grid model [Bai et al., 2006] were not able to model the vehicular mobility adequately, because mobility was only constrained to a set of grid-like streets. Further, advancing of the mobility models was achieved by using map generation techniques, such as Voronoi graphs [Davies et al., 2006], which constrain the movement of the vehicles to a network of irregular streets generated artificially. Recently, the most prominent mobility models started utilizing real world maps (e.g., Choffnes and Bustamante [2005] and Mangharam et al. [2005]).

The vehicle interaction includes modeling the behavior of a vehicle that is a direct consequence of the interaction with the other vehicles on the road. This includes the microscopic aspects, such as lane changing and decreasing/increasing the speed due to the surrounding traffic, as well as the macroscopic aspects, such as, taking a different route due to the traffic conditions, for instance, congestion [Gipps, 1986].

With regards to the implementation approaches for the microscopic mobility models, Car following models (CFMs) are probably the most popular class

of driver model. CFMs usually represent time, position, speed, and even acceleration as continuous functions, but most have been extended to provide discrete formulations [Hartenstein et al., 2010]. The most prolific proponents are [Helbing, 2001]: the cellular automata models [Nagel and Schreckenberg, 1992], the follow-the-leader models (e.g., car-following [Rothery, 1992] and intelligent driver model (IDM) [Kesting et al., 2010], the gas-kinetic models [Hoogendoorn and Bovy, 2001] and the macroscopic models [Lighthill and Whitham, 1955]. These analytical models are important to this work, specifically the IDM, because they are used to represent movement in our modeling of the Sections 5.3 and 7.1.

The IDM shows a crash-free collective dynamics, it exhibits controllable stability properties and implements an intelligent braking strategy with smooth transitions between acceleration and deceleration behavior [Kesting et al., 2010]. The IDM acceleration is a continuous function incorporating different driving modes for all speeds in traffic of highways or cities. It uses the current speed v and the distance s (bumper-to-bumper) to the leader vehicle ($s = x_l - x - e$, where x are the coordinates and e the extent of vehicle). This method also takes into account the speed difference (approaching rate) $\Delta v = v - v_l$ to the leading vehicle. The IDM acceleration function is given by

$$a_{IDM}(s, v, \Delta v) = a \left[1 - \left(\frac{v}{v_0} \right)^\delta - \left(\frac{s^*(v, \Delta v)}{s} \right)^2 \right] \quad (3.1)$$

$$s^*(v, \Delta v) = s_0 + vT + \frac{v\Delta v}{2\sqrt{ab}} \quad (3.2)$$

This expression combines the free-road acceleration strategy $a_{free}(v) = a[1 - (v/v_0)^\delta]$ with a deceleration one $a_{brake}(s, v, \Delta v) = -a(s^*/s)^2$, which becomes relevant when the gap to the leading vehicle is not significantly larger than the effective ‘desired (safe) gap’ $s^*(v, \Delta v)$. The free acceleration is characterized by the desired speed v_0 , the maximum acceleration a , and the exponent δ characterizing how the acceleration decreases with velocity ($\delta = 1$ corresponds to a linear decrease while $\delta \rightarrow \infty$ denotes a constant acceleration). The effective minimum gap s^* is composed of the minimum distance s_0 , the ‘velocity dependent distance’ vT , which corresponds the speed of the leader vehicle with a constant ‘desired time gap’ T , and a dynamic contribution, which is only active in non-stationary traffic corresponding to situations in which $\Delta v \neq 0$. This latter contribution implements an ‘intelligent’ driving behavior that, in normal situations, limits braking decelerations to the comfortable deceleration b . In critical situations, however, the IDM deceleration becomes significantly higher, making the

IDM collision-free [Treiber et al., 2000]. The IDM parameters v_0, T, s_0, a and b are shown in Table 3.1.

Table 3.1: Parameters of the IDModel. Adapted from Kesting et al. [2010]

Parameter	Car	Truck
Desired speed v_0	120 km/h	85 km/h
Free acceleration exponent δ	4	4
Desired time gap T	1.5	2.0
Jam distance s_0	2.0	4.0
Maximum acceleration a	1.4 m/s^2	0.7 m/s^2
Desired deceleration b	2.0 m/s^2	2.0 m/s^2

New position and speed at time t , can be given respectively, by:

$$x = x_i + vt + \frac{a}{2}t^2 \quad (3.3)$$

$$v = v_i + at \quad (3.4)$$

The deceleration over a given distance may be calculated by the Torricelle equation:

$$v^2 = v_i^2 + 2a\Delta x \quad (3.5)$$

where v is the current speed, v_i e a are respectively, initial speed and acceleration, finally, Δx is the gap position between vehicle and obstacle.

3.2.1 Lane-Changing Model - Model MOBIL

A general model to represent lane-changing rules was proposed for Dirk Helbing and Martin [2007]. The model is called Minimizing Overall Braking Induced by Lane Change (MOBIL). The ‘‘utility’’ and the ‘‘risk’’ associated of a given lane are determined in terms of longitudinal accelerations calculated by microscopic car- following models as IDM. The behind vehicle desacceleration in the target lane can not exceed a given safe limit b_{safe} . *Risk criterion* prevents critical lane changes and collisions, the *incentive criterion* takes into account the advantages and disadvantages of other drivers associated with a lane change via ‘‘politeness factor’’. The parameter allows one to vary the motivation for lane changing from purely egoistic to more cooperative driving behavior.

To understand this model, a specific lane change is shown in Figure 3.3. The MOBIL model depends generally on the two behind vehicles in the current and the target lanes, respectively. Thus, for a vehicle c considering a lane change, the behind vehicles in the target and current lanes are represented by n and o . The acceleration a_c denotes the acceleration of vehicle c on the actual lane, and \tilde{a}_c refers to the situation in the target lane, that is, to the new acceleration of vehicle c in the target lane. Likewise, \tilde{a}_o and \tilde{a}_n denote the acceleration of the old and new behind cars after the lane change of vehicle c [Dirk Helbing and Martin, 2007].

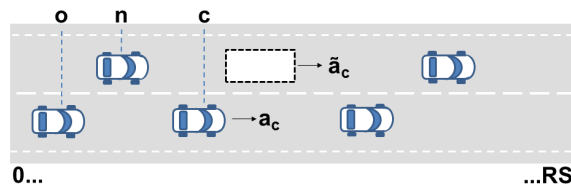


Figure 3.3: Mobil notations - Font [Dirk Helbing and Martin, 2007]

3.2.1.1 Incentive Criterion for Symmetric Lane-Changing Rules

According to Dirk Helbing and Martin [2007], incentive criterion normally determines if the lane change is better or not for an individual driver. In this model, the incentive is generalized to include the immediately affected neighbors. The politeness factor p determines to which degree these vehicles influence the lane-changing decision. Thus, the incentive criterion is given by:

$$\underbrace{\tilde{a}_c - a_c}_{\text{driver}} + p \left(\underbrace{\tilde{a}_n - a_n}_{\text{new behind}} + \underbrace{\tilde{a}_o - a_o}_{\text{old behind}} \right) > \Delta a_{th} \quad (3.6)$$

The first two terms denote the advantage of a possible lane change for the driver. The change is good if the driver can go faster in the new lane. The third term denotes the total advantage of the two immediately affected neighbors multiplied by politeness factor p . Finally, the Δa_{th} term on the right-hand side of Equation 3.6 models a certain inertia and prevents lane changes if the overall advantage is only marginal compared with a “keep lane” directive.

If necessary, more than two lanes per direction can be implemented. For example, a vehicle in a center lane, the incentive criterion is satisfied for both neighboring lanes, the change is performed to the lane in which the incentive is larger. The p values can vary from 0 (for selfish) to 1 (altruistic drivers). In the last case, drives do not change if the overall traffic situation would be worst. By setting $p < 0$, even malicious drivers

could be modeled, who accept own disadvantages in order to thwart others. In the special case $p = 1$ and $\Delta a_{th} = 0$, the incentive criterion is simplified to:

$$\tilde{a}_c + \tilde{a}_n + \tilde{a}_o > a_c + a_n + a_o \quad (3.7)$$

Thus, lane changes are only performed when they increase the sum of accelerations of all involved vehicles, which corresponds to the concept of minimizing overall braking induced by lane changes (MOBIL) in the ideal sense [Dirk Helbing and Martin, 2007].

3.2.1.2 Incentive Criterion for Asymmetric Passing Rules

In most countries, the driving rules for lane usage are restricted by legislation. A common policy is to adopt the right lane as default. Thus, an asymmetric lane-changing criterion for two-lane freeways also was formulated. Specifically, the following European traffic rules are assumed [Dirk Helbing and Martin, 2007]:

1. **Passing rule:** Passing in the right-hand lane is forbidden unless traffic flow is congested. Any vehicle driving at a velocity below some suitably specified velocity v_{crit} is treated as driving in congested traffic (e.g., $v_{crit} = 30$ km/h).
2. **Lane usage rule:** The right lane is the default lane. The left lane should only be used for the purpose of overtaking.

The keep-right directive of the lane usage rule is implemented by a constant bias Δa_{bias} in addition to the threshold Δa_{th} . Therefore, the resulting asymmetric incentive criterion for lane changes from left (L) to right (R) is [Dirk Helbing and Martin, 2007]:

$$L \rightarrow R : \tilde{a}_c^{eur} - a_c + p(\tilde{a}_o - a_o) > \Delta a_{th} - \Delta a_{bias} \quad (3.8)$$

The incentive criterion for a lane change from right to left is given by

$$R \rightarrow L : a_c - \tilde{a}_c^{eur} + p(\tilde{a}_n - a_n) > \Delta a_{th} + \Delta a_{bias} \quad (3.9)$$

The passing rule was implemented following the condition:

$$\tilde{a}_c^{eur} = \begin{cases} \min(\tilde{a}_c, a_c) & \text{if } v_c > \tilde{v}_{lead} > v_{crit} \\ a_c, & \text{otherwise} \end{cases} \quad (3.10)$$

Where, according to Dirk Helbing and Martin [2007], \tilde{a}_c corresponds to the acceleration in the left lane and \tilde{v}_{lead} denotes the velocity of the front vehicle in the

left-hand lane. The passing rule influences the acceleration in the right-hand lane only if (a) there is no congested traffic ($\tilde{v}_{lead} > \tilde{v}_{crit}$), (b) the front vehicle on the left-hand lane is slower ($\tilde{v}_c > \tilde{v}_{lead}$), and (c) the acceleration \tilde{a}_c for following this vehicle would be lower than the single-lane acceleration a_c in the actual situation. It should be noted that the condition $v_c > \tilde{v}_{lead}$ prevents vehicles in the right-hand lane from braking whenever they are passed. The parameter values used in the MOBIL simulations are shown in the Table 3.2.

Table 3.2: Parameters of the Mobil Model. Font– [Dirk Helbing and Martin, 2007]

Parameter	Value
Politeness factor p	0...1
Changing threshold Δa_{th}	0.1 m/s^2
Maximum safe deceleration b_{safe}	4 m/s^2
Bias for right lane Δa_{bias}	0.3 m/s^2

The politeness parameter p , which it is relative to the incentive criterion, mainly determines the lane-changing rate. The changing threshold Δa_{th} prevents lane changes of marginal advantage. For $p < 1$, the maximum safe deceleration b serves as an additional safety criterion. The value of b is chosen considerably below the physically possible maximum deceleration of about 9 m/s^2 on dry roads. In the case of asymmetric (European) lane-changing rules, the additional bias Δa_{bias} models a preferred lane usage of the default lane.

3.2.2 Framework for Realistic Vehicular Mobility Models

For the purpose of guiding developers through various challenges and options for modeling vehicular motions, Boban and Vinhoza [2011] proposed a concept map for a comprehensible representation of the functionalities of a realistic vehicular mobility model. Figure 3.4, the concept map is organized around two major modules: the motion constraints and the traffic generator. Additional modules such as time and external influences are also required for a fine tuning of the mobility patterns. The framework description is as follows [Boban and Vinhoza, 2011]:

- **Motion constraints** describe the relative degree of freedom available to each vehicle. They can be streets, buildings, neighboring cars, pedestrians.
- **Traffic generator** defines different kinds of vehicles and it deals with their interactions according to the environment under study. Macroscopically, it models traffic densities, speeds and flows, while microscopically it deals with

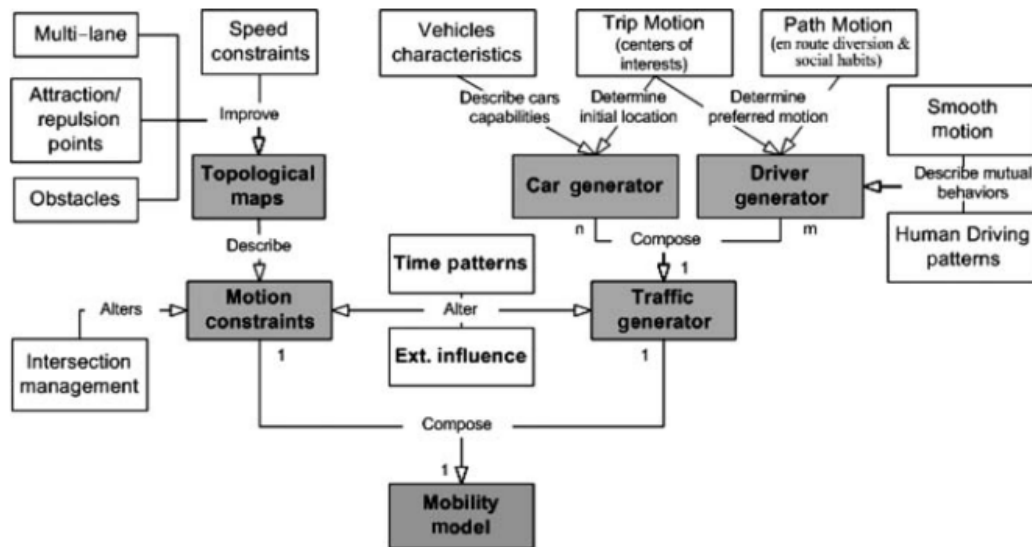


Figure 3.4: Concept map for realistic mobility models – Font [Boban and Vinhoza, 2011]

properties such as the inter-distance between cars, acceleration, braking, and overtaking

- **Time** describes different mobility configurations for a specific time of the day. Traffic density is indeed not identical during the day. Peak times, such as rush hours or during special events can be observed. This block influences the motion constraints and the traffic generator functional blocks.
- **External influences** model the impact of a communication protocol or any other source of information on the motion patterns. This block also models the impact of accidents, temporary road works, or real-time knowledge of the traffic status on the motion constraints and the traffic generator blocks.

3.3 Data exchange (Network) model

Network models for VANETs are quite similar to those used in other fields of MANET research. The data models used in the current simulators, such as NS-2 and NCTU-NS [Wang and Chou, 2008], rely on discrete event simulation, where different protocols of the network stack are executed through events triggered by upper layer. The main difference arises in the use of a dedicated WAVE protocol stack. The lowest layers have been standardized under IEEE 802.11p specification. On the network layer,

WAVE Short Message Protocol (WSMP) is used. However, applications running over the standard TCP/IP protocol stack are also supported.

Due to the relative novelty of WAVE protocols, the majority of the widely used VANET simulators do not implement these resources. One exception is the NCTU-NS simulation environment [Wang and Chou, 2008]. Modeling the network stack realistically is important for the credibility of the results obtained at each level of the protocol stack, and especially for the application level, since all the potential simulation errors from the lower layers are reflected at the application layer [Boban and Vinhoza, 2011].

3.4 Radio-propagation model

Appropriate models have been developed in order to represent adequately the signal propagation in vehicular networks. They regard the unique characteristics of VANET environment, for instance, high speed of the vehicles, obstruction-rich setting and the specific location of the antennas. Overall, we can distinguish two class, the deterministic and the stochastic model (Figure 3.1). Deterministic models attempt to model the signal behavior based on the exact environment in which the vehicle is currently located and with specific locations of the objects surrounding the vehicle [Maurer et al., 2004]. Stochastic models, on the other hand, assume a location of the surrounding objects based on a certain (often pre-defined) statistical distribution [Acosta-Marum and Ingram, 2006].

Stochastic models use the concepts of computational geometry to characterize the environment by generating the possible paths or rays between the vehicle of transmitting and receiving. Some examples are presented by Nagel and Eichler [2008] and Giordano et al. [2010]. An interesting research is proposed by Zuniga and Krishnamachari [2004]. The link probability is calculated within arbitrary networks with regard to the distance between nodes and their relative clustering. To understand this method the Figure 3.5 will be used. We can assume that the sender k is closest to receiver i , so its signal is the strongest; Sender j is the weakest. All link probabilities are affected by the others activities. Here d , d' and d'' are the distances from the senders to the receiver i .

The node j wants to send a message to i . The probability that i receives j 's message is computed as a function of the signal-to-noise ratio, $SNR_{i,j}$, which is given for $SNR_{i,j} = rx_{i,j}/bgN_{i,j}$. It is the ratio of the power of the received message at i ($rx_{i,j}$), and the noise ($bgN_{i,j}$) generated in part by the other activities of the network [Zuniga

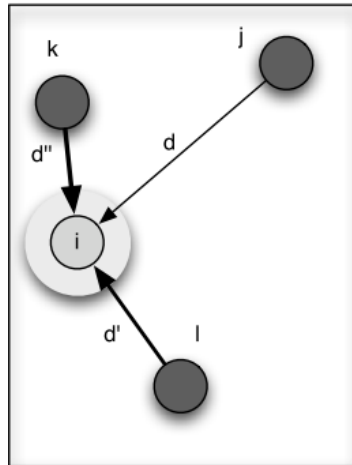


Figure 3.5: Signal strength varying with distance and interference - Adapted by [Boulis et al., 2008]

and Krishnamachari, 2004]. The power at the receiver can now be computed directly:

$$rx_{i,j} = 10^{rxdB_{i,j}/10} \quad (3.11)$$

Where, $rxdB_{i,j}$ is signal strength of the received message. It depends on the distance $d(i,j)$ between i and j , and the power at which j transmits, tx_j , and is given by the formula:

$$rxdB_{i,j} = tx_j - PLd_0 - 10(pLE)\log_{10}(d(i,j)/10) \quad (3.12)$$

where pLE is called the Path Loss Exponent, and can be thought of as the rate at which the signal strength deteriorates with distance. Finally, d_0 and PL are scaling constants determined by the environment.

Next, we compute the background noise ($bgN_{i,j}$). Thus, the noise generated by the other nodes in the network must be taken into account (See Figure 3.5). Let $send_k$ be a function which is 1 or 0 according to whether node k is transmitting a message or not. The total background noise at receiver i interfering with the message transmitted by j is given by [Zuniga and Krishnamachari, 2004]:

$$bgN_{i,j} = nbgN + \sum_{k \neq i,j} rx_{i,k} * send_k \quad (3.13)$$

In the simple case, where there are no neighbors nodes, the background noise is assumed to be a constant $nbgN$ determined by the operating environment. Now the two quantities $bgN_{i,j}$ and $rx_{i,j}$ given at 3.11 and 3.13 respectively we can now compute the probability that i receives j 's message.

According to Boulis et al. [2008], the current analytic models for computing the link probabilities predict that there is signal-to-noise threshold which there is effectively zero probability that the message can be decoded by the receiver. This threshold depends on a number of network specific parameters: the data rate (nDR), the noise bandwidth (nBW), the threshold probability (nTP), the frame length (f) of the message, and the modulation type of the transmission. In Frequency Shift Keying (FSK) modulation, the threshold is given by:

$$\Delta_{i,j} = -2 \frac{nDR}{nBW} \log_e(2(1 - nTP^{\frac{1}{sf}})) \quad (3.14)$$

By Equation 3.15, Zuniga and Krishnamachari [2004] computed the threshold-free probability that j's message is received by i:

$$snr2prob(SNR_{i,j}) = (1 - 0.5 * \exp(-0.5 \frac{nBW}{nDR} SNR_{i,j}))^{sf} \quad (3.15)$$

With $SNR_{i,j}$ and the $snr2prob$ we can compute the link probabilities

$$prob_recv_{i,j} = \begin{cases} 0 & \text{if } SNR_{i,j} < \Delta_{i,j} \\ snr2prob(SNR_{i,j}), & \text{otherwise} \end{cases} \quad (3.16)$$

According to Boulis et al. [2008], since this formula depends on the mutual contribution to the noise of the surrounding nodes, this is actually a conditional probability, namely the probability that i receives j's message given that i does not receive the message from any of the other nodes. This together with the assumption that if $j \neq k$ then the events "i receives a message from node j" and "i receives a message from node k" are mutually disjoint, implies that we can compute the probability P_i that any message is received by i (from whichever sender) as the sum of the individual link probabilities:

$$P_i = \sum_{j \neq i} prob_recv_{i,j} * send_j \quad (3.17)$$

According to Correia [2011] another renowned statistical model to represent fading of the radio-mobile signal is the model of Nakagami-m [Nakagami, 1960], who is indicated by Taliwal et al. [2004]; Rubio et al. [2007] as an excellent model to represent the VANET environment. Its probability density function (pdf) is given by:

$$f(x) = \frac{2m^m x^{2m-1}}{\Gamma(m)\Omega^m} \exp\left[-\frac{mx^2}{\Omega}\right], \quad x \geq 0, \quad \Omega > 0, \quad m \geq \frac{1}{2} \quad (3.18)$$

where, Ω is the expected value of the distribution and can be interpreted as the average received power. m is the fading parameter. The values of parameters m e Ω are in functions of the distance and Γ represents the Gamma distribution.

Due to the complexity of the exact model for probability density functions, approximate solutions have been developed. Thus, studies [Jang and Sichitiu, 2012; Malone et al., 2007; Engelstad, Paal E. and Osterb, Olav N., 2005] use a geometric approach to reducing the computational effort and simplify the calculations of PDF and CDF distribution Nakagami-m. An interesting example is shown in Killat and Hartenstein [2009], they present an analytical model that gives the probability of successfully receiving an one-hop broadcast message based on the distance between sender and receiver. They obtained the expected probability of successfully receiving a message at distance d while considering an intended communication range of CR meter:

$$Pr(d, CR) = \exp^{-3(d/CR)^2} \left(1 + 3 \left(\frac{d}{CR} \right)^2 + \frac{9}{2} \left(\frac{d}{CR} \right)^4 \right) \quad (3.19)$$

Both analytical models presented here are important to this work. They are used to represent signal propagation in our modeling of the Sections 5.3 and 7.1.

3.5 Discussion

Up to this moment, the chapter presented a description of the current VANET simulation. We notice a constant evolution to obtain a realistic environment to execute testing in Intelligent Traffic Systems. There are several software options with different levels of granularity in mobility, network and signal propagation models. However the search for improving the verification should continue. Most traffic and network models still use the Isolated communication approach. The software that makes use of Federated approach suffers with latency requirements, due synchronization among models. This last method is computationally complex, because both simulators must run simultaneously [Hartenstein et al., 2010]. Another problem is that many network simulators do not support the WAVE protocol [Boban and Vinhoza, 2011].

Although the interaction of vehicles and the application of traffic rules have been shown to be essential for the accuracy of the vehicular traffic model [Harri et al., 2009], many mobility models for VANET have scarce support for these microscopic aspects [Ferreira et al., 2009]. It must be considered also that current VANET simulators consider vehicles as dimensionless entities, in other words, have no influence

on the signal propagation [Martinez et al., 2011].

In the remainder of this chapter, we present some of the background on Virtual traffic lights (VTL). This done to verify the statements made above regarding simulation, in the other words, we want to show how simulation have been employed in VANETs. Thus, we introduce concepts, goals, tests and requirements about VTL. We describe four different environments and protocols of the virtual traffic lights proposed in the literature. Finally, a qualitative evaluation of safety in VTL protocols is presented.

3.6 A Survey on Virtual Traffic Lights

Traffic congestion has become one of the main problems in modern life. Several policies have been adopted worldwide to minimize this problem, and for over thirty years now, efforts have been made to create traffic light systems that can respond to the ever increasing traffic.

Currently most traffic control systems rely on timing plans generated offline by traffic engineers using optimization models. However, these systems are hard to maintain and do not adapt to special traffic events [Gradinescu et al., 2007], such as rush hours.

Another problem is the cost of maintenance. According to DoT [2007], there is a close relationship between costs and traffic volume – the USA spends billions of dollars to maintain its current traffic lights. However, the low traffic volume at certain intersections are unavoidable and it might not be worth using traffic lights in these situations. For instance, only one percent of the intersections in the USA are managed by traffic lights [Bureau, 2013; Patrushev, 2008], despite the fact that several studies suggest that semaphores can reduce accidents by up to 33% [DoT, 2004].

Nevertheless, the management of intersections has gained attention recently and technological breakthroughs in mobile computing and wireless communication have created opportunities for intelligent semaphores. The main idea behind new traffic lights is that vehicles at intersections would elect a leader based on a set of rules. Thus, the leader creates a Virtual Traffic Light (VTL), which it will control the traffic flow by broadcast of messages. In order to establish a VTL, the algorithm can consider different types of information, e.g. the current state of the traffic flow, which is shared by all vehicles, and could potentially improve the traffic flow [Santos et al., 2010].

There are several goals that can be considered in the design of a VTL protocol [Ferreira et al., 2010; Gradinescu et al., 2007; Chou et al., 2013]:

1. Reducing the number of accidents at intersections
2. Reducing the travel time of urban workers during rush hours
3. Increasing the energy efficiency of urban transportation
4. Mitigating traffic congestion
5. Increasing the productivity of a nation
6. Reducing the carbon emission of vehicles
7. Reducing the deployment and maintenance costs of traffic lights
8. Support a greener environment by reducing external infrastructures

The analysis of VTL has been focused on two major areas: (i) quantitative benefits over traditional methods and (ii) analysis of traffic performance metrics, such as overall gain in average speed, reduction of carbon dioxide emission, and improvements in traffic flow at intersections.

However, most studies overlook the quality of VTL communication, which can become very poor due to several reasons. For instance, conditions when there is no line-of-sight (LOS) between vehicles due to obstacles, such as buildings, that are often not considered. These situations can cause major failures in the protocol and, therefore, incur in potential accidents and the loss of human lives. Thus, in order to address these issues, researchers have to improve the understanding of these protocols.

3.7 Virtual Traffic Lights (VTL)

Virtual Traffic Lights (VTL) can be divided in two different types:

1. built with mobile ad hoc networks (MANETs) and its derivatives
2. based on loop detectors (fixed sensors)

For the latter, we can cite the projects SCOOT [Robertson and Bretherton, 1991] and SCATS [Akcelik and Chung, 1998]. SCOOT is based on loop detectors placed in every lane on an intersection, usually in the end. Other systems, including SCATS, have detectors placed immediately before the stop line at an intersection. Nevertheless, they cannot get accurate data when the queue grows beyond the length of the detector, or the lane is over saturated. Since they use a model based especially on occupancy, they also have difficulties in differentiating between high flows or paralyzed intersection.

However, adaptive traffic lights based on wireless communications among vehicles can employ more flexibility than the ones mentioned above, because they provide more information for the decision process (e.g. vehicles positions and speeds). The following subsections describe some of these methods.

3.7.1 Adaptive VTL Using Car-to-Car Communication

The work, presented by Gradinescu et al. [2007] examines the possibility of deploying an adaptive signal control system in intersections. The system can base its decision control on information coming from cars. The authors assume each vehicle is equipped with a short-range wireless communication device and there is a controller node placed in the intersection with traffic lights.

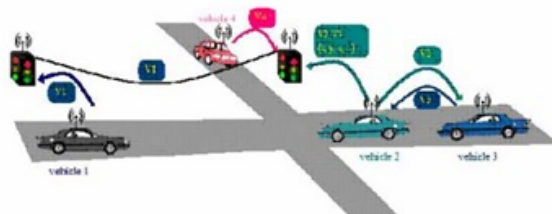


Figure 3.6: Traffic lights communicate with adapt timing – Font [Gradinescu et al., 2007]

This work uses TrafficView [Dashtinezhad et al., 2004], a VANET project for data dissemination among vehicles. By making use of wireless communication and GPS, it enables vehicles to collect and disseminate traffic information. Vehicles periodically transmit information about themselves and other cars, which they know about. They use one-hop transmission to avoid a broadcast storm. Each record consists of a position, identification number, speed, direction, state and a timestamp of the moment when the information was created.

The Controller receives all the exchanged information, thus finding out how crowded the intersection lanes are. Beyond, in an urban environment, controllers in adjacent intersections may communicate through a network. This can provide each other with additional information [Gradinescu et al., 2007]. For example, with a known number of cars approaching, the signal timing can be optimized. This model is depicted in Figure 3.6.

The Traffic Lights Controller keeps track of the vehicles throughout the entire period when they are in a few miles range around the intersection (through the information propagation scheme of TrafficView), so it is able to measure accurately both volume and demand. A time optimization plan occurs every cycle and is valid for

the next cycle. This is done based on the measured parameters. But during a cycle new optimizations may happen.

The authors have used simulation to analyze the behavior of the proposed method. They emphasize the difficulty of integration between traffic and wireless network simulators. Thus, they have chosen to develop a VANET discrete-event simulation tool joining mobility and computer network. An integrated module for computing the fuel consumption and pollutant emissions was also coupled. They have focused on the relation between fuel consumption and the speed and acceleration of the vehicle. The proposed method found that the system significantly improves traffic fluency, compared to the existing, pre-timed traffic lights.

3.7.2 Adaptive VTL Based on VANETS for Mitigating Congestion in Smart City

Chou et al. [2013] proposed an approach which adapts the cycle of VTL according to current traffic and vehicle type using VANETS. The expected contribution is to mitigate congestion. The proposed VTL dynamically adapts the cycle of traffic lights according to three vehicle types, such as light vehicles, buses, and emergency ones. In this method each vehicle is equipped with an on-board unit (OBU), the drivers should obey the traffic signals of the proposed method. The routing protocol uses technique of carry and forward [Vasilakos et al., 2011], in which nodes accept data from a source and carry the message until transmitting it. The architecture of proposed VTL contains three stages (See Figure 3.7a), called Smart Cycle Function, which is described below [Chou et al., 2013]:

Planning function – vehicles collect *beacon messages* from their neighbors for the selection of leader candidates. The beacon messages contain instant speed, location, travel time, next-intersection, moving direction, vehicle identity, vehicle type and generating time. Each lane should have a candidate and the vehicle that has the shortest travel time until the intersection is selected. Then, the selected leader candidate broadcasts *lane messages* to other vehicles and other leader candidates. The *lane messages* contain vehicle identity, travel time, next-intersection, lane identity, average speed of neighbors, number of neighbors and generating time.

Management function – In this phase *lane messages* are received. The leader selection is made multiplying the quantity by weight assigned according to three types of vehicles in every lane. Then, a leader candidate that has the smallest weight is selected as leader. If the selected leader has not enough safe stopping distance, a leader candidate that has the second smallest weight is chosen, and so on. The selected

leader creates and maintains its own VTL and each intersection has only one leader. For example, in Figure 3.7b there are two leader candidates, A and B. The weight of vehicles in A, on vertical road segment is smaller than the weight of B, on horizontal road segment. Therefore, leader candidate A is selected as leader, and A's traffic signal is set as red. The selected leader has to broadcast *VTL messages* to other vehicles that are in the transmission range. The *VTL messages* contain VTL type, intersection identity, traffic signal, cycle time and generating time.

Operating function – vehicles receive traffic signals (VTL messages) broadcast by the selected leader. The traffic signals are shown on the screen of OBU. If vehicles do not receive any VTL messages, vehicles can go through intersections and periodically send beacon messages.

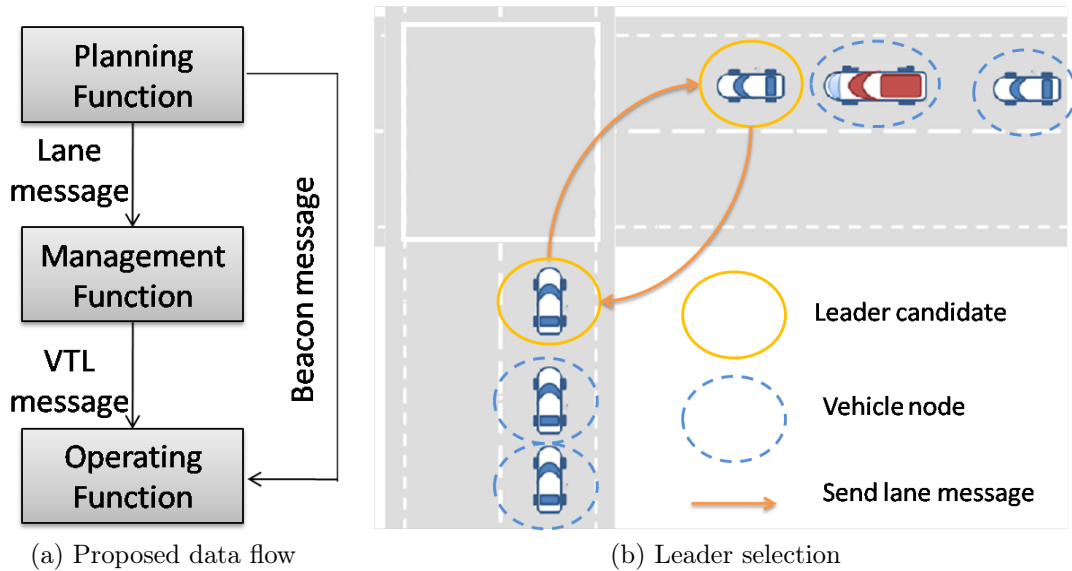


Figure 3.7: AVTL scheme – Font [Chou et al., 2013]

In order to check the method, the researchers used the VANET simulator called NCTUns [Wang and Chou, 2008]. NCTUns provides OBU and SignalAgent modules. OBU module can exchange information with each other. SignalAgent module was used to control traffic signal according to real time information.

Simulation parameters were as follows: IEEE 802.11p is MAC protocol, and its transmission range is 250 meters. The transmission frequency of beacon message, lane message and VTL message is set as 100 milliseconds. The road length of each two-way road segment was 500 meters, and there were 4 lanes. Different numbers and types of vehicles were used. When the number of vehicles is 200, the proposed VTL approach improves the average speed of all vehicles by 33.1% when compared with fixed cycle

traffic light. The average speed of emergency vehicles is improved by 62.06% when simulated with 800 vehicles.

3.7.3 Distributed VTL System

Santos et al. [2010] present a distributed VTL system which is based on crossroad traffic conditions. VTLs are generated by a distributed algorithm which requires participation from vehicles at different roads in a particular intersection. Every vehicle has a component called Car Navigation Modules (CNM) to display and change informations. In a real scenario, communication among vehicles is made by wireless, which works in a broadcast fashion.

The goal of this algorithm is to allow the communication among vehicles in order to control the traffic at intersections using Virtual Traffic Lights. For this, the authors defined some spatial variables. The Figure 3.8 shows them. The Line Of No Return (LONR) defines the point when the vehicle must take a decision about if it should stop or go ahead. It must stop if there are vehicles in the other roads or go ahead if no other vehicles are detected. The Line Of Activity (LOA) is the point at which the algorithm exits the *Idle state* and becomes active, preparing for VTL negotiation. The algorithm assumes that the distance between intersections is bigger than the radius of LONR.

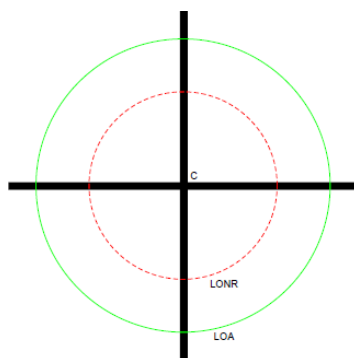


Figure 3.8: Crossroad with LOA and LONR lines depicted – Font [Santos et al., 2010]

Figure 3.9 presents a state machine of the algorithm. The vehicles periodically send beacons (messages) to announce their presence. When vehicle closes the intersection and crosses the LOA line it starts proposing himself as leader. It sends a periodic beacon, the proposing beacons, moving from Idle state to Propose state. By the time it arrives at LONR line if had received a beacon from an existing Leader or a better proposing beacon from other close car then goes to a Non Leader state. Otherwise the vehicle goes to Leader state and starts sending leader beacons. If there

are no vehicles in the concurrent roads of the current intersection then the vehicles jump to Yellow state and the vehicle can cross the intersection with caution.

If more than one road is occupied then there will be several leaders, one for each occupied road. These leaders, which represent their roads, must then decide between themselves who will cross the intersection. For that, leaders start to negotiate to choose one among them to manage the crossroad, the *SuperLeader*. To achieve that, after they make sure that they are seeing each other, they jump to the *SuperLeader* Negotiation state where the negotiation takes place. The winner jumps to *SuperLeader* state and the others return to the Leader state. The *SuperLeader* starts sending periodic green beacons to the target road for a period of time. The vehicles that receive the green message jump to Green state and can start moving. After the time expires, the *SuperLeader* returns to Leader state and the negotiation starts again [Santos et al., 2010].

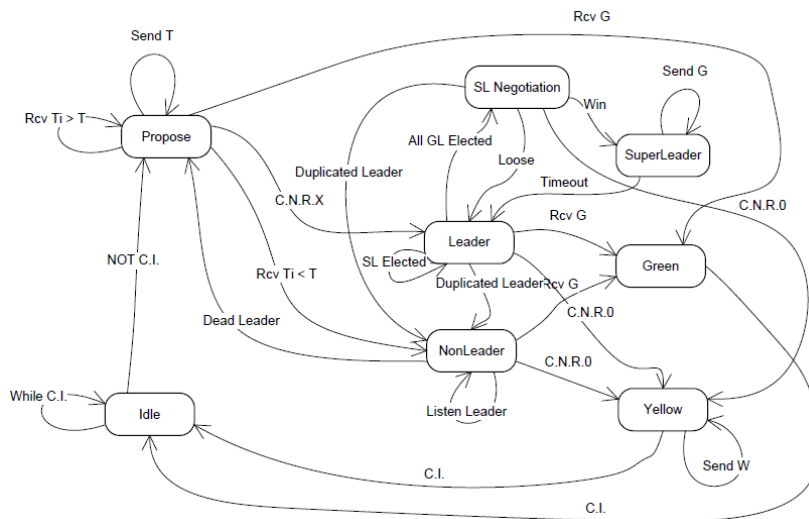


Figure 3.9: Distributed VTL State Machine – Font [Santos et al., 2010]

In VTL algorithm different messages are used for different purposes. The messages have common fields like source, destination, timestamp and type. However, there are specific information for each purpose. For example, Traffic Messages carries traffic information per each lane of a road. VTL Messages are used by leader negotiation. These messages are the following ones [Santos et al., 2010]:

- **Propose:** Contains the propose time of the sending CNM. The CNM with the lower propose time for given road becomes the leader of that road;
- **Leader:** Contains propose time and fellow leaders, which is the number of leaders from where this CNM is receiving Leader messages. The other field in this

message is a hash calculated from the IDs of each leader CNM from where this CNM is receiving messages;

- **Negotiation:** This message contains the same fields as the previous one, plus the number of cars in this CNM Leader group (this message is only sent by Leader CNMs) and the SLV, that is value associated with each Leader and which is then used to choose who will be the Super Leader and which road will receive green semaphore;
- **Green (G):** Contains the same information as the original VTL message plus the road ID to where the green semaphore is destined and the GPS time at which this green semaphore will end;
- **Yellow (W):** Has the same information as the original VTL message plus the GPS time the first white message was sent. This value is used to ensure fairness;
- **Black:** Signals a car mal-function, e.g., wireless device is broken.

Validation tests have not been published by authors. However, they have been implementing a simulation framework in order to test and validate their VTL algorithm. A simulation infrastructure have developed. This infrastructure is composed by several entities, which are the Traffic Simulator (TS), the Manager, Car Navigation Modules (CNMs) and Network Simulator (NS). CNM is the module that real vehicles would have in order to enable the VTL algorithm. It provides communication between different vehicles. This communication allows vehicles to discover each other and to exchange information necessary to the VTL algorithm [Santos et al., 2010]. Java language was the choice to implement all modules with exception of TS and CNM Display. The navigation module is implemented in Python. The traffic simulator was already implemented in C++. Instead of creating a Traffic Simulator from scratch, the authors opted out by modifying an already existent simulator, adapting it to their needs. Due to its performance, and modularity, DIVERT [Conceição et al., 2008] was chosen.

3.7.4 Self-Organized Traffic Control

Ferreira et al. [2010] present a VTL system which is based on the assumptions that all vehicles are equipped with WAVE devices, share the digital map and have a global positioning system (GPS). Each vehicle has a dedicated computer called Application Unit (AU) which maintains an internal database. The data are grouped in a *location table*, containing information about every node in its vicinity. These information are

constantly updated through the reception of new beacons. The other table is named *traffic signs*, it is responsible for storing data about the traffic light configuration at an intersection. The AU also display information to the driver.

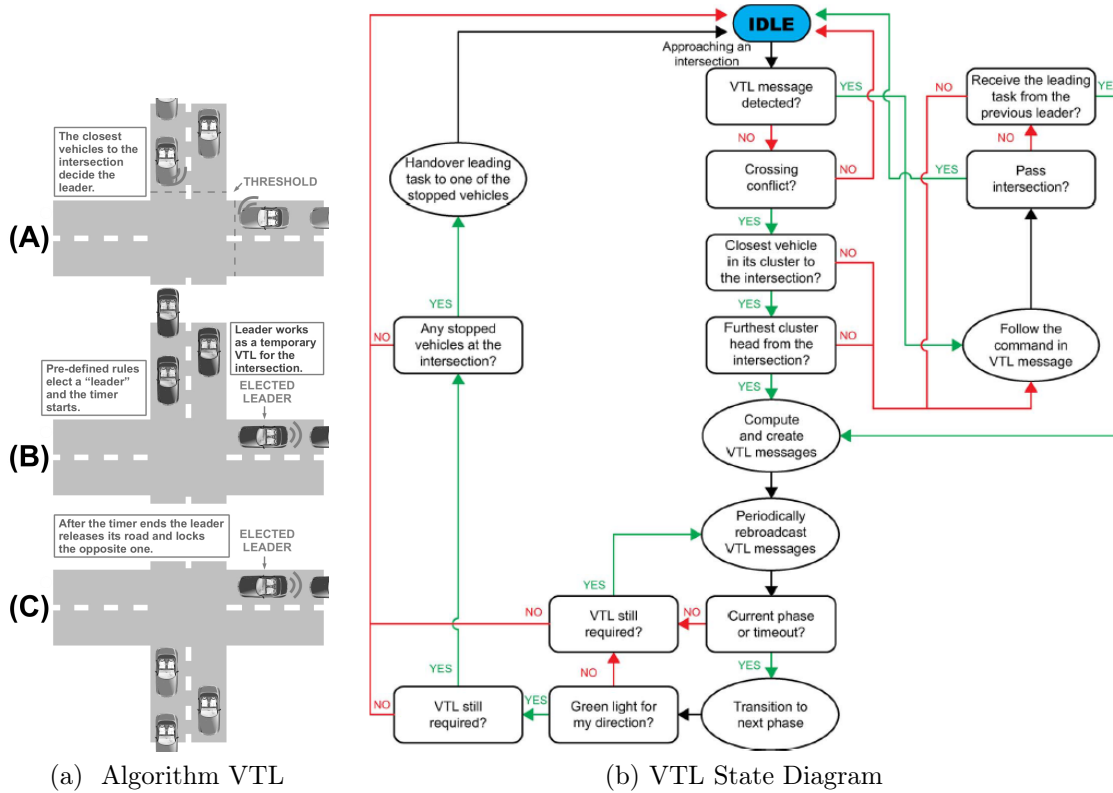


Figure 3.10: VTL scheme – Font [Ferreira et al., 2010]

The VTL operation principle is simple (illustrated in Figure 3.10a). **Step 1:** vehicles reaching an intersection check if there is a VTL running to be obeyed or if one needs to be created in order to avoid crossroad conflicts. **Step 2:** during the VTL creation, all vehicles nearing the intersection must elect one vehicle, which will be responsible for broadcasting the traffic signal messages. **Step 3:** once there is a leader, a VTL cycle for the intersection control is initiated with a red light for the leader lane. This condition ensures that the leader will remain in the intersection for the duration of a complete cycle. Once the current phase is finished, a new leader must be elected to maintain the VTL. Figure 3.10b depicts the principle of VTL operation in terms of states.

Figure 3.11a shows an example of the envisioned in-vehicle traffic light, where the information are on the windshield of the car, such, the driver can easily see what to do when approaching an intersection. This in-vehicle equipment will be interfaced with the DSRC radio [Tonguz, 2011].

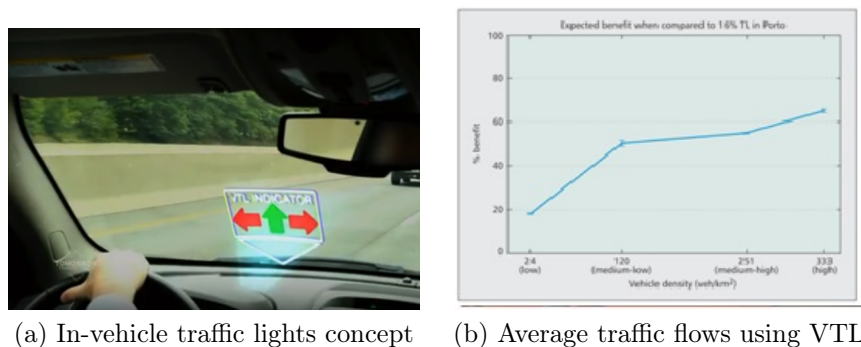


Figure 3.11: VTL – Font [Tonguz, 2011]

To see the impact of the proposed method, the DIVERT simulator was used to analyze the traffic in the Manhattan grid and in city of Porto in Portugal, which comprises 965 km of road structure and 2000 intersections of which 328 are equipped with traffic lights (16 percent of all intersections) [Ferreira et al., 2010]. This software is able to do evaluation in large-scale of micro-simulation [Conceição et al., 2008]. The DIVERT mobility model has been validated against empirical data collected through a recently conducted comprehensive stereoscopic aerial survey [Ferreira et al., 2009].

Figure 3.11b shows the results of a large-scale implementation of the proposed VTL scheme with four scenarios of vehicle densities. The proposed scheme leads to about 60 percent increase in average flow rates in Porto at high vehicle densities (i.e., during rush hours) [Ferreira et al., 2010]. The percentage benefit is with respect to the existing traffic management scheme. The same group presents in [Ferreira and d'Orey, 2012], results referent a Carbon Dioxide (CO_2) emissions which show a significant reduction on CO_2 when using VTLs, reaching nearly 20% under high-density traffic.

However, according to Tonguz [2011], despite the good results, there are several challenges that need to be carefully addressed. The leader election algorithm described earlier has to be made fail-safe. For instance, there is radio frequency propagation problems due to the existing buildings and/or other obstructions. They might cause malfunctioning of the protocol resulting in accidents.

3.8 Qualitative Evaluation of Safety in VTL Protocols

Use of VTL has been shown to be promising and the works presented above show gains in traffic flow of up to 60%. However, those studies were based on the assumption of a perfectly reliable communication, i.e., notification messages of the traffic light were

always received by vehicles located within a certain distance to the sender. Hence, effects such as signal fading or non-line-of-sight conditions due to buildings were neglected. Such effects, however, can have a negative impact on the dissemination of the notification messages [Neudecker et al., 2012]. Virtual traffic lights suffers from the same time-critical requirements as well as all safety applications.

In Mangel et al. [2011], the authors assert that adverse conditions exist at most intersections in urban environments. Therefore, the usage of ITS may add a new safety risk if a robust and reliable operation is not guaranteed. For instance, if individual vehicles do not receive red light notifications early enough in a VTL, a car-crash or dangerous driving maneuvers may be the consequence.

Worried about reliability in communication VTLs, Neudecker et al. [2012] analyzed a scenario where, the speed and distance for two vehicles crossing an intersection are exactly the same (see Figure 3.12). The algorithm examined was the presented in Ferreira et al. [2010] what is described above. Hence, effects such as signal fading or non-line-of-sight (NLOS) conditions due to buildings were studied.

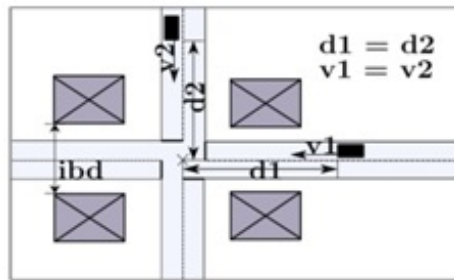


Figure 3.12: Layout scenario test – Font [Neudecker et al., 2012]

Radio propagation was modeled using the NLOS communication model proposed by Mangel et al. [2011]. The evaluation was based on the following metrics [Neudecker et al., 2012]:

- VTL leader election distance (m): the distance at which the coordination of the VTL protocol is finished and a vehicle declares itself as the VTL leader.
- Required deceleration (m/s^2): this metric can be directly deduced from the distance at which a driver is signalized to stop and the vehicle's speed.

According to the results shown in Neudecker et al. [2012], the expected impact of NLOS conditions on the performance of VTL can be confirmed. However, the results indicate that this impact is not significant and that NLOS conditions do not prohibit a timely detection and the possibility to take appropriate actions. Despite

these results show the feasibility of VTLs under NLOS channel conditions, the authors claim that further research is needed to cover all the possible scenarios and other sources of impairment. One of the problems is that, simulation examines only a subset of possible behaviors.

3.9 Conclusions

Vehicular Networks (VANETs) increasingly draw interest in academic and commercial sectors. There are several potential applications for it, from entertainment to the prevention of accidents. But it is essential to test and analyze VANETs in order to prevent loss of life. Simulation is widely used to check new protocols and applications.

For VANET simulations, microscopic and macroscopic models are typically required. The first provides space-time behavior of vehicles and their interactions on an individual level. The macroscopic model can show average number of vehicles per hour passing a specific cross-section, the average number of vehicles per kilometer, etc.

These models started with random movements in a plane area and nowadays they progressed to real maps. However, despite the advance, significant challenges must be overcome. For instance, the great benefit of VANET is to transmit information about the traffic, in order to modify the routes of vehicles. However, modeling this benefit has become a challenge for the simulation. Thus, two hitherto unconnected worlds must now work together, network and traffic simulators.

Currently, the vehicular traffic flow simulators can produce trace files that are given as input to a network simulator, or the traffic flow simulator must be coupled with the network simulator to allow feedback from communication to vehicular traffic behavior. Another technique is the use of a software to do the interface between established Network and Mobility simulators. In this context, there are challenges that must be addressed by the research community [Hartenstein et al., 2010; Hartenstein and Laberteaux, 2008; Boban and Vinhoza, 2011; Alves and et al., 2009]: (1) Specifications of APIs for coupling traffic flow and networking simulators (2) Modeling how drivers react to the additional information provided by VANETs (3) Definitions of benchmark to make simulation studies and results comparable (4) Defining the required level of accuracy in simulation according to application type. (5) Most simulators do not properly represent the hardware and protocols of vehicular networks (e.g. Wave protocol and modern chipsets) (6) Modeling and analyzing the effect of large scale fading, in particular of moving radio-wave obstacles like a truck between two cars.(7)

Real-world measurements show that deterministic radio propagation models should be avoided because they do not capture the probabilistic effects of small scale fading that have a significant impact on packet reception.

Virtual traffic lights is one of many types of applications which use simulations to perform tests. The analysis in these applications are important by the fact that the usage is directly related to human lives. VTL usually are implemented in a distributed manner by vehicles themselves, and allow to self-adapt the signaling schedule with respect to the current traffic volume.

The study about VTL was important for this work in order to show the types of test are applied and the challenges found in this phase. The conclusions show their evaluations are based on optimistic and ideal assumptions, i.e., they have neglected the existence of radio obstacles and considered a perfect communications system.

Tables 3.3 and 3.4 showed a comparative among the VTLs presented in this chapter. The first one depicts the employed technology, who it indicates a trend to use ad hoc communication. The last table presents the types of made analysis, which it displays the focus on evaluation of large scale traffic efficiency. It is unusual investigate the feasibility like in the work presented in Section 3.8.

Table 3.3: Analysis of VTL for infrastructure and in-vehicle features.

	Infrastructure	In-Vehicle Features
1	Intersection-based	Short-range wireless communication device, GPS
2	Vehicle-based	On-board unit (short-range wireless communication device, GPS and e-maps)
3	Vehicle-based	Car Navigation Modules (short-range wireless communication device, GPS)
4	Vehicle-based	DSRC devices vehicles sharing the same digital road map, GPS

Table 3.4: Analysis of VTL for measure of effectiveness and tests.

	Measure of Effectiveness	Tests
1	Control delay, volume per capacity ratio and pollutants emission	A simulator was created for the experiments
2	Average speed	NCTUns as network simulator tool (It provides OBU and SignalAgent modules)
3	Validation tests haven't been published yet	Network simulator was created to be integrated to the DIVERT
4	Flow rate vehicle/ km^2 and pollutants emission	DIVERT traffic simulator who has a simple network embedded

The researchers also have difficulty to execute tests because the simulation joins two fields not integrated before (traffic and network computer). Thus, most works have to build their own simulators or implementing an interface able to do the integration among both.

Chapter 4

Probabilistic Model Checking

Outline. In this chapter we present some of the background on model checking that is relevant to this thesis. We also describe probabilistic model checking and the PRISM tool. Their probabilistic logic and reward-based extensions are also presented. Finally we discuss about the related works to this thesis and a conclusion is exposed.

4.1 Introduction

The model checking technique is a formal method to automatic and exhaustively analyze if a given model of a system respects a given specification.

The systems usually modeled are hardware, communication standards or software, such as circuit designs, network protocols (e.g. Carrier Sense, Multiple Access with Collision Detection) or critical software (e.g. aircraft or spacecraft), respectively. Recently other applications have been considered, for instance, biological systems, and more recently game theory.

The specification is often given in special types of logic, for example, temporal and probabilistic logic, which allow defining properties about the sequence and probability of certain events, respectively. The specification often contains safety requirements such as the absence of deadlocks and critical states that can cause the system to crash.

The first section presents the symbolic version of model checking, along with its related concepts such as binary decision diagrams and temporal logics. This section is largely based on [Clarke et al., 1999] and [Song, 2004].

The probabilistic version called probabilistic model checking (PMC) is described in the following section, including its special representation using multi-terminal binary decision diagrams and more appropriate probabilistic logic. This section is largely based on [Parker, 2002], [Braz et al., 2013a] and [Crepalde, 2011].

PMC is used in this research through PRISM model checker to study movements of vehicles and their interactions with the environment. Therefore, a brief description of PRISM and its modeling and property languages was included.

Finally, modeling about ad hoc networks are illustrated, the aim is to describe the weakness and strengths about related work.

4.2 Symbolic Model Checking

This technique was proposed independently and simultaneously by Edmund M. Clarke and Ernest A. Emerson [Emerson and Clarke, 1980; Clarke and Emerson, 1982; Clarke et al., 1986] and by Jean P. Queille and Joseph Sifakis [Queille and Sifakis, 1982]. In 2007, Clarke, Emerson and Sifakis shared the Turing Award for their contribution on founding the field of model checking.

The systems are modeled as a finite state machines, described in a precise high level modeling language, and properties are specified in temporal logics. Given a model M , a set of initial states S_0 and a property ϕ , the verification algorithm automatically checks if the model respects the property ϕ . This is performed by exhaustively exploring the transitions between states, checking the specified properties.

There are other methods such as tests and simulations to analyze and check systems. However, the model checking technique offers several advantages. First, it is completely automatic: after modeling the system and specifying the desired properties, the model checker performs the analysis without human interaction.

Furthermore, model checking guarantees that the model respects the specified properties since the state space is completely searched. This allows the detection of even small errors which might pass unnoticed by other techniques such as emulation, simulation and tests. Finally, if the model does not respect some given property, the model checker usually produces a counter-example, which is useful to understand and correct the error.

However, there are some limitations. This exhaustive exploration of the state-space causes the classical model checking problem of the state space explosion. The number of states grows exponentially with the number of variables. For example, the composition of N variables of size k each yields k^N states. This is an active research topic, and several efforts have been made to reduce the state space. The first and one of the most important contributions has been made in [McMillan, 1992], which proposed using Binary Decision Diagrams (BDDs), originally created by [Bryant, 1986], to symbolically represent the transition relations between states.

This implicit representation encodes each state as the attribution of boolean values to the system variables. Therefore, transitions can be expressed as boolean formulas in terms of two sets of variables, one set encoding the previous state and another set encoding the current state. This avoids the explicit construction of graphs of system states, providing a more compact description of the model, increasing the size of models from 10^5 to 10^{20} states.

Furthermore, several improvements have been made to cope with the state space explosion, allowing the verification of systems with 10^{120} states, such as Partial Order Reduction [Godefroid et al., 1996], Symmetry Reduction [Clarke et al., 1998], Compositional Reasoning [Berezin et al., 1998], Statistical or Approximate Model Checking [Younes, 2005; Clarke et al., 2008] and Bisimulation Minimisation [C. Dehnert and Parker, 2013].

4.2.1 Kripke Structure

The representation used in symbolic model checking to capture the behavior of a system is a directed state transition graph called **Kripke structure**. It is a variation of a non-deterministic finite state machine whose states or nodes represent the reachable states of the system and whose edges represent allowed transitions between states.

Atomic propositions of the model are boolean expressions over the variables, constants and other symbols of model. These propositions assume the value of either true or false. Atomic propositions are self contained and do not include other propositions.

Let AP be the set of atomic propositions. A Kripke structure M over AP is a 4-tuple $M = \langle S, I, R, L \rangle$, as defined by [Clarke et al., 1999], where:

- S is a finite set of states.
- $I \subseteq S$ is the set of initial states.
- $R \subseteq (S \times S)$ is a transition relation that must be total, i.e., for each state $s \in S$, there is a state $s' \in S$ such that $R(s, s')$.
- $L : S \rightarrow 2^{AP}$ is a function that labels each state with the set of atomic propositions which are true in that state.

A *path* in a Kripke structure M from a state s_0 is an infinite sequence of states $\pi = s_0 s_1 s_2 \dots$ such that $s_0 = s$ and the relation $R(s_i, s_{i+1})$ holds for all $i \geq 0$.

The Figure 4.1 shows a graphical representation of a Kripke structure for a model and a computational path in its structure. The components of M which define the Kripke structure are:

- $S : \{S_0, S_1\}$
- $I : \{S_0\}$
- $R : \{(S_0, S_1), (S_1, S_0), (S_1, S_1)\}$
- $L(S_0) = \{A, B\}, L(S_1) = \{A, \neg B\}$

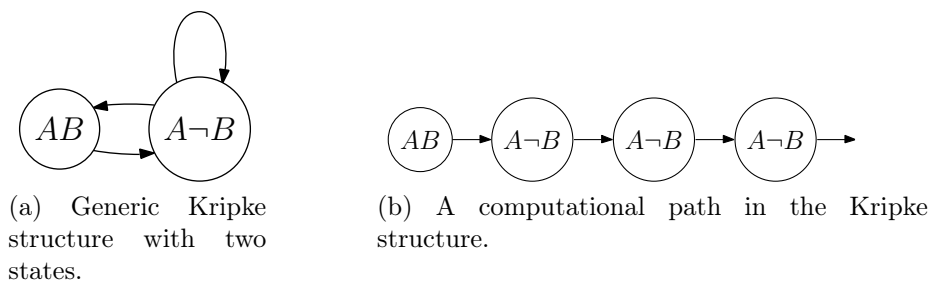


Figure 4.1: A Kripke structure and a computational path in it.

4.2.2 Kripke Structure Representation

As described by Clarke et al. [1999], a Kripke structure can be represented through boolean functions, which employ logical operators such as negation (\neg), disjunction (\vee), conjunction (\wedge) and implication (\implies).

Let $V = \{v_1, \dots, v_n\}$ be the finite set of **system variables** and $D = \{D_1, \dots, D_n\}$ be the finite set of **domains** of these variables, such as D_i represents the set of possible values for v_i . A **system state** is the evaluation of all system variables at a specific instant in time.

For example, suppose a system with three boolean system variables (a three bit counter), i.e., let $V = \{v_1, v_2, v_3\}$ and $D_1 = D_2 = D_3 = \{0, 1\}$. There are eight possible system states ($2^3 = 8$), such as $(v_1 = 0, v_2 = 0, v_3 = 0)$, $(v_1 = 0, v_2 = 0, v_3 = 1)$ and $(v_1 = 0, v_2 = 1, v_3 = 1)$. Their respective representations as boolean functions are $(\neg v_1 \wedge \neg v_2 \wedge \neg v_3)$, $(\neg v_1 \wedge \neg v_2 \wedge v_3)$ and $(\neg v_1 \wedge v_2 \wedge v_3)$, where v_i is an 1-valued variable (or true) and $\neg v_i$ is a 0-valued variable (or false). These representations are **symbolic**, therefore the name symbolic model checking.

The transitions between system states must also be represented as boolean functions. In order to do that, a second set of system variables is created to represent

the system variables in the next (future) state. Therefore, V is the set of system variables in the current state, and V' is the set of system variables in the next (future) state. For each $v \in V$, a variable $v' \in V'$ for the next state is created. A transition can be viewed as an ordered pair for the evaluation of system variables in V and V' , which can be represented as a conjunction of boolean functions.

Let f be the boolean function for the current system state s and f' be the boolean function for the next (future) system state s' , then the transition from state s to state s' is represented by the conjunction of both boolean functions, therefore, $f \wedge f'$. For example, the transition from a state where $(v_1 = 0, v_2 = 0, v_3 = 0)$ to a state where $(v_1 = 0, v_2 = 0, v_3 = 1)$ is represented by the boolean function $(\neg v_1 \wedge \neg v_2 \wedge \neg v_3) \wedge (\neg v'_1 \wedge \neg v'_2 \wedge v'_3)$, as shown in Figure 4.2.

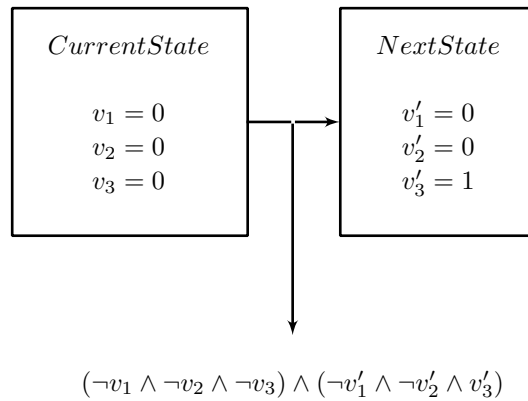


Figure 4.2: The symbolic representation of a transition.

Boolean functions can represent a set of states and a set of transitions. If f_1, f_2, \dots, f_n represent all the transitions of a Kripke structure, the boolean function for the set of all transitions is described by the disjunction of all f_i , i.e., $f_R = f_1 \vee f_2 \vee \dots \vee f_n$. The same reasoning is used to make the set of all system states of a Kripke structure.

The coupling of several transitions into a simple boolean function, which simplifies the process of graph traversal, is inherent to the representation of boolean functions as BDDs, covered in the next section. This is one of the main reasons for the efficiency of BDDs in symbolic model checking algorithms.

Furthermore, the set of atomic propositions AP must be described in order to create specifications for the system. An atomic proposition is an expression of the form $v = d$, where $v \in V$ and $d \in D$. A proposition $v = d$ will be true in a system state s , if the boolean function of s becomes true when v assumes the value d .

In order to illustrate first order representations, a symbolic representation of a Kripke structure is presented in Figure 4.1. In this example, let $V = \{A, B\}$ and

$D_A = D_B = \{0, 1\}$ for the model. Furthermore, it is necessary to create two variables $V' = \{A', B'\}$ to represent future states. The symbolic representations of the system states s_0 and s_1 are given by the boolean functions $A \wedge B$, and $A \wedge \neg B$, respectively. Finally, the transition from state s_0 to state s_1 is given by $R(s_0, s_1) \equiv A \wedge B \wedge A' \wedge \neg B'$.

The boolean function which represents the complete transition relation of the model is composed of three disjunctions, representing the number of transitions of the Kripke structure:

$$(A \wedge B \wedge A' \wedge \neg B') \vee (A \wedge \neg B \wedge A' \wedge B') \vee (A \wedge \neg B \wedge A' \wedge \neg B')$$

In this example, the labeling function L contains the following mappings: $L(s_0) = \{A = 1, B = 1\}$ and $L(s_1) = \{A = 1, B = 0\}$. As $A = 1$ and $B = 1 \in L(s_0)$, when A and B assume, respectively, the values 1 and 1 in system state s_0 , the boolean function which represents this state ($A \wedge B$) becomes true.

Although previous definitions have considered only a boolean domain $D = \{0, 1\}$ for all variables, it is possible to use other domains such as integer values by simply encoding each element to a boolean domain. The encoding is performed by using j bits to encode the domain D_i of each variable v_i as a binary number and represent v_i as j boolean variables.

For example, suppose that the variables v_i assume integer values in the domain $D = \{0, 1, 2, 3, 4, 5, 6, 7\}$, then only three bits are necessary to encode each variable v_i . Therefore, the atomic proposition ($v_1 = 2$) can be represented by the conjunction of three new boolean variables ($\neg v_{1.1} \wedge v_{1.2} \wedge \neg v_{1.3}$), where each one corresponds to a bit of the binary codification of the integer value of 2 (010). Finally, the boolean function $(v_1 = 2) \wedge (v_2 = 3) \wedge (v_3 = 4)$ would be represented in a binary domain as shown below:

$$(\neg v_{1.1} \wedge v_{1.2} \wedge \neg v_{1.3}) \wedge (\neg v_{2.1} \wedge v_{2.2} \wedge v_{2.3}) \wedge (v_{3.1} \wedge \neg v_{3.2} \wedge \neg v_{3.3})$$

4.2.3 Binary Decision Diagrams

A data structure called Binary Decision Diagram (BDD) is used to represent boolean functions in a compact, efficient and canonical form, as first described in [Bryant, 1986]. A BDD is compact because it discards redundant information; efficient because it allows rapid graph traversal; and canonical because it is easy to check if two boolean functions are equivalent.

This data structure is often used in symbolic model checking to represent finite states systems, although there are other representations, such as explicit-state

representation (all variables are always represented) and conjunctive normal form (a conjunction of disjunctions).

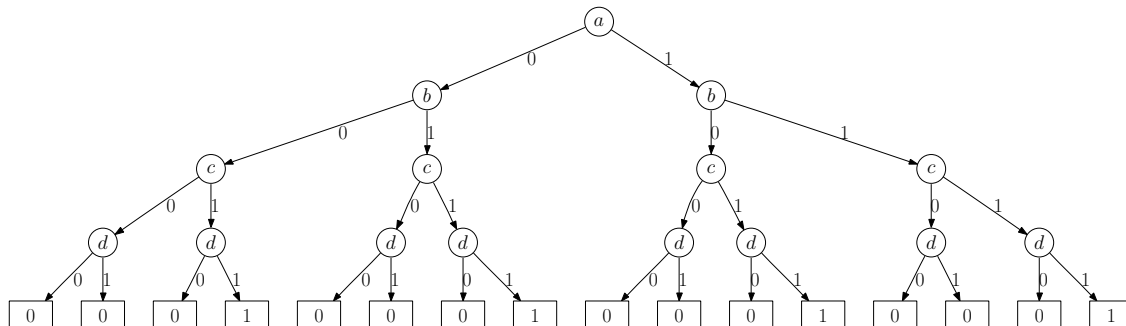


Figure 4.3: A binary decision tree for the boolean function $(a \wedge b) \vee (c \wedge d)$.

BDDs represent boolean functions as a special type of binary decision trees, which is a directed tree with two types of vertices: terminal and non-terminal vertices (shortened as terminal and non-terminal, respectively). In a binary decision tree, a non-terminal vertex v is labeled with a boolean variable, given by $var(v)$, which has two successors: $zero(v)$, when $var(v) = 0$, and $one(v)$, when $var(v) = 1$. A terminal vertex v assumes only the value zero or one (simply zero- or one-terminal, respectively), given by function $value(v)$. The tree edges are labeled with the value zero or one.

The Figure 4.3 shows an example of a binary decision tree for the boolean function $(a \wedge b) \vee (c \wedge d)$. A path in the tree starts at the root vertex and the choice between $zero(v)$ and $one(v)$ directs to a terminal that represents the function evaluation. In the example, the variable evaluation $(a = 1, b = 0, c = 1, d = 0)$ leads to a zero-terminal, which means that the boolean function is false for that assignment, while a different assignment leading to a one-terminal means that the function is true.

The binary decision trees can have a lot of redundant information. For example, in the Figure 4.3 there are eight subtrees whose roots are labeled with a boolean function d . However, only three of them are unique: in the first unique subtree, assigning any value to the variable d leads to a zero-terminal. On the other hand, in the second unique subtree, assignment any value to d leads to a one-terminal. Finally, in the third unique subtree, the assignment of zero to d leads to a zero-terminal, while assigning one to d leads to a one-terminal.

Therefore, a BDD is obtained by merging identical subtrees and eliminating nodes with repeated information. The resulting structure is a directed acyclic graph (DAG) called BDD which, unlike trees, allow shared vertices and substructures. It is worth to emphasize the canonical aspect of BDDs, which means that two functions are equal if,

and only if, their associated BDDs are isomorphic¹.

[Bryant, 1986] first showed how to obtain a canonical representation of boolean functions by imposing two restrictions on BDDs. Foremost, the variables must appear in the BDD in the same order along the path from its root to a terminal. Secondly, isomorphic subtrees or vertices should not exist in the BDD.

The first restriction is satisfied by fixing an order for the variables which label the vertices of the BDD ($var_1 < var_2 < \dots < var_n$). This means that if a vertex u precedes a non-terminal v (starting from the BDD root), then the variable which labels the vertex u precedes the variable which labels v in the order ($var(u) < var(v)$).

The second restriction is satisfied by repeatedly applying three rules of transformation that do not change the function represented by the BDD:

1. Remove duplicated terminals: keep only two terminals, one zero-terminal and another one-terminal. Redirect all input edges from the removed terminals to these two unique terminals;
2. Remove duplicated non-terminals: if two non-terminals u and v are labeled with the same variable ($var(u) = var(v)$) and have the same successors ($zero(u) = zero(v) \wedge one(u) = one(v)$), remove the vertex v and redirect its input edges to the vertex u ;
3. Remove redundant children: if a non-terminal v has $zero(v) = one(v)$, then remove v and redirect all of its input vertices to $zero(v)$.

The canonical form of the BDD, obtained by imposing these two restrictions, is called Ordered Binary Decision Diagram (OBDD). The Figure 4.4 shows the OBDD for the binary decision tree of 4.3, considering the variable order $a < b < c < d$.

Although BDDs have several advantages, it has its own disadvantages. The main one is the order of the variables which appear in the boolean function being represented. Depending on it, the BDD can be heavily compressed or completely redundant. However, the problem of choosing the variable ordering which minimizes the BDD size is co-NP-complete [Bryant, 1986].

There are heuristics to approximate this problem, such as the one presented by [Bollig and Wegener, 1996], whom also briefly reviews several other heuristics based on local search and simulated annealing. Nonetheless, variable orderings are often found empirically since one can expect that it is related to the semantics of the model.

¹Two BDDs are isomorphic if there is an injective function h that maps terminals and non-terminal from a BDD to the other.

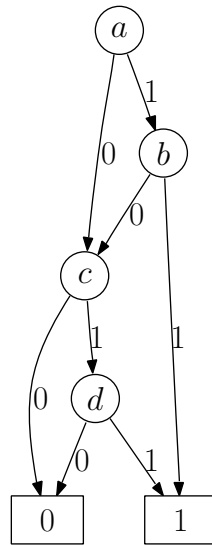


Figure 4.4: A reduced binary decision diagram.

Another issue with BDDs is the space complexity. Since BDDs are essentially an exhaustive representation of the model, in the worst case it is exponential to the number of variables of the boolean function [Bryant, 1986].

4.2.4 Temporal Logic

There are several different types of logic, from the classical and first mathematical formalism created by George Boole, and named after him, the boolean algebra, to other non-orthodox logics, such as the more modern modal logic, the epistemic logic, or the logic of knowledge. These logics are appropriate to different types of systems.

However, reactive and parallel systems present additional challenges on their reasoning. They often can not be understood from its current state, demanding the analysis of a sequence of events. For example, one simple reactive system such as an alternating bit communication protocol can be reasoned only using temporal properties, such as “if a message is sent, it is eventually received”. The message can be received in the next time unit, or in the next ten time units. Therefore, to reason about such systems we need to be able to state temporal properties.

One could describe and reason logical propositions in terms of time and the sequence of events, which is called temporal logic and its applied to model checking, where the behavior of the system being modeled is represented as a state-transition graph such as the BDD reasoning on the time evolution of the model.

There are several temporal logics, such as the Computation Tree Logic (CTL and CTL*) and Linear Time Logic (LTL). Each logic has its own set of logical operators.

The main difference between LTL and CTL, is that in LTL the reasoning is on an infinite computational path, while in CTL the reasoning is on a tree of infinite paths starting at a root node s_0 . The CTL* is a superset of CTL and LTL.

Temporal logics (TL) borrow quantifiers from predicate logic (or first-order logic), such as for all (\forall) and exists (\exists), naming them path quantifiers, which allow reasoning on the computational tree created from unfolding the Kripke structure (Figure 4.5). They also introduce temporal operators, which allow reasoning on the sequence of states. TL can also use logical operators from boolean algebra, such as negation (\neg), disjunction (\vee), conjunction (\wedge) and implication (\implies).

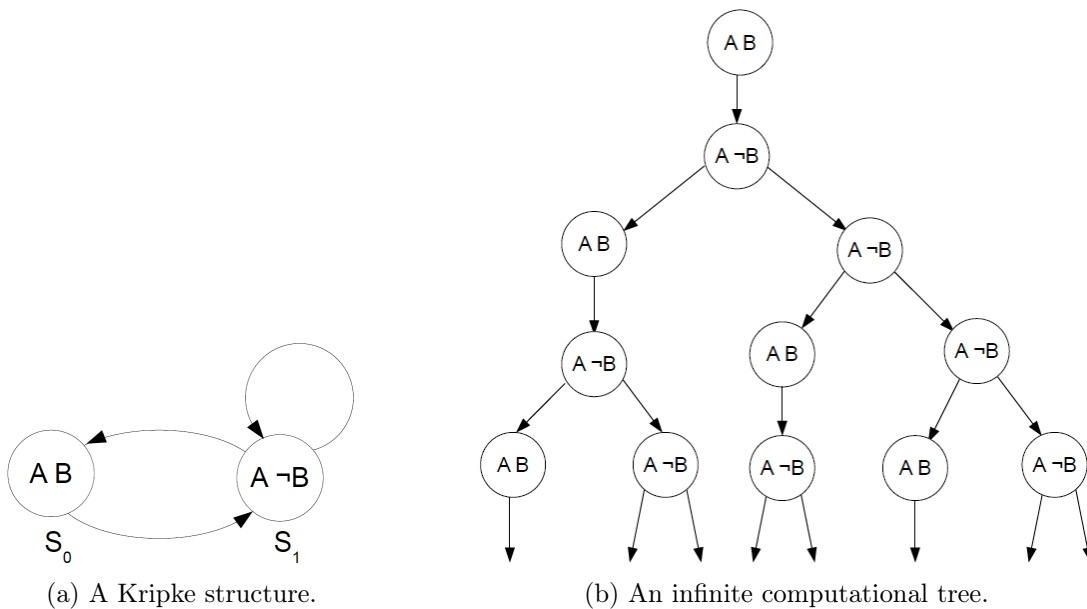


Figure 4.5: Unfolding a Kripke structure in an infinite computational tree.

Path quantifiers reason on a computational path, or sequence of states. They are used to specify that every computational path from the current state respects the given property. Path quantifiers are shown below. LTL does not support the **E** path quantifier, because there is a single computational path.

- The “For All” path quantifier: **A** Φ – for every path, Φ is true.
- The “Exists” path quantifier: **E** Φ – there exists a path where Φ is true.

Temporal operators reason on a sequence of states. They are used to check that one or more states hold the given property. The Figures below show the computational paths associated with each property. The black dot represents the state in which the

property ϕ is true. Only for Figure 4.9, the black and dashed dots represent when the properties ϕ_1 and ϕ_2 are true, respectively.

- The “Eventually” operator: $\mathbf{F} \phi$ – ϕ is true in a future state (Figure 4.6).

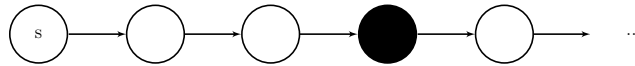


Figure 4.6: The “Eventually” operator: $\mathbf{F} \phi$.

- The “Globally” operator: $\mathbf{G} \phi$ – ϕ is true in all future states (Figure 4.7).

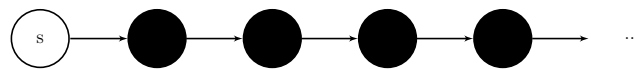


Figure 4.7: The “Globally” operator: $\mathbf{G} \phi$.

- The “Next” operator: $\mathbf{X} \phi$ – ϕ is true in the next state (Figure 4.8).

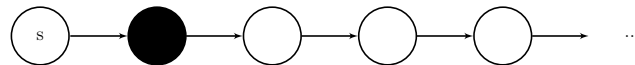


Figure 4.8: The “Next” operator: $\mathbf{X} \phi$.

- The “Until” operator: $\phi_1 \mathbf{U} \phi_2$ – ϕ_1 is true until ϕ_2 becomes true (Figure 4.9).

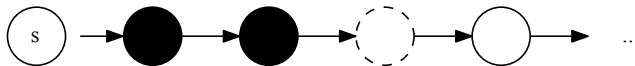


Figure 4.9: The “Until” operator: $\phi_1 \mathbf{U} \phi_2$.

In CTL, temporal operators must be preceded by a path quantifier, for example, $\mathbf{EG} \phi$, which states that exists a path where ϕ is always true (Figure 4.10d). The most commonly used CTL operators are shown below (Figure 4.10).

Property $\mathbf{AF} \phi$ checks if a property ϕ is eventually observed in all paths starting at the current state (Figure 4.10a). Property $\mathbf{EF} \phi$, shown in Figure 4.10b, checks if exists a path where the property ϕ is eventually observed. Property $\mathbf{AG} \phi$ checks if a property ϕ is always observed in all paths starting at the current state (Figure 4.10c). These properties could be used to check safety aspects of the model, such as situations which should never occur, for example, “the system never crashes!” ($\mathbf{AG} \text{!}crash$).

There are several other logic, such as temporal or probabilistic logic. They can be applied in probabilistic model checking which it is a formal verification technique to study systems that exhibit probabilistic behavior. In the next Section we will discuss this type of formal verification.

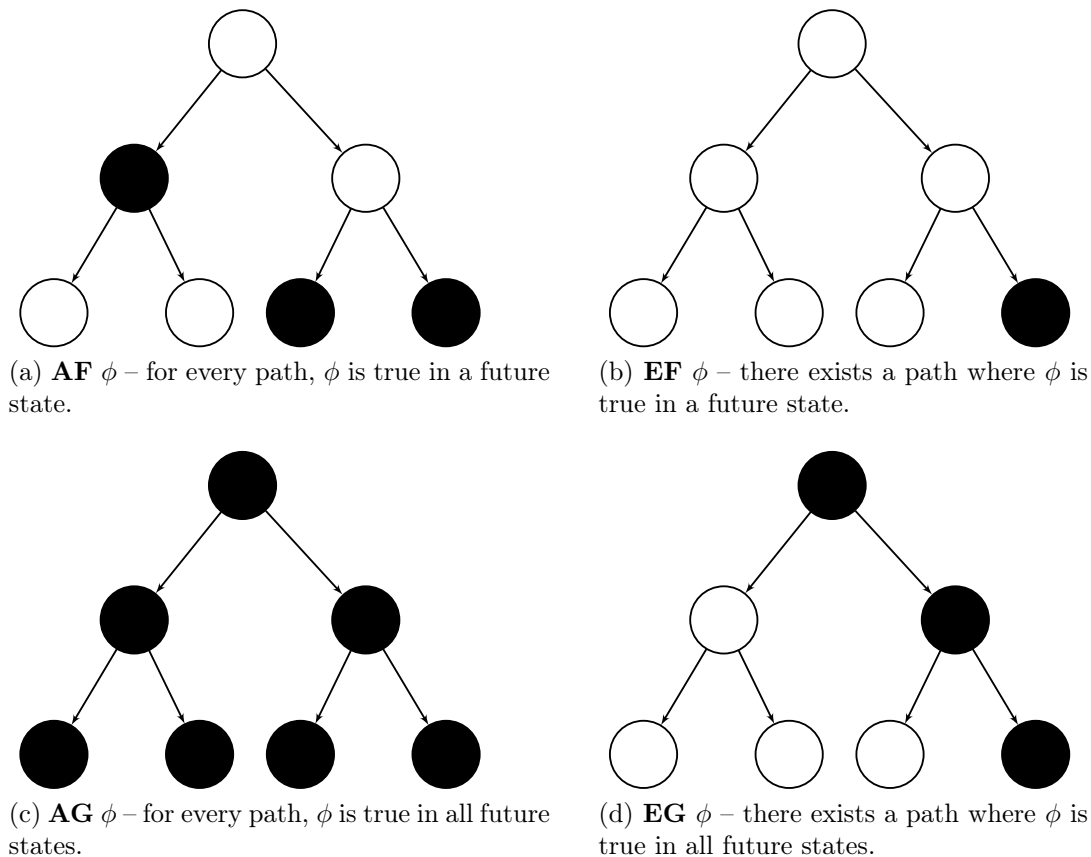


Figure 4.10: Different types of combinations of temporal operators.

4.3 Probabilistic Model Checking

Probabilistic Model Checking (PMC) is a formal, exhaustive and automatic technique for modeling and analyzing stochastic systems. PMC checks if a model satisfies a set of properties given in special type of logic. This method can supply several types of probabilistic models: probabilistic timed automata or discrete- and continuous-time Markov chains and Markov decision processes.

A stochastic system M is usually a Markov chain or a Markov decision process. This means that the system satisfies the Markov property, i.e., its behavior depends only on its current state and not on the whole system history, and each transition between states occurs in real-time.

Given a property ϕ expressed as a formula in a probabilistic temporal logic, PMC attempts to check whether a model of a stochastic system M satisfies the property ϕ with a probability greater than or equal to a probability threshold $\theta \in [0, 1]$.

Tools called models checkers such as PRISM [Kwiatkowska et al., 2011] attempt to check the models. It requires two inputs: a modeling description of the system, which

defines its behavior (for example, through the PRISM language), and a probabilistic temporal logic specification of a set of desired properties (ϕ).

The model checker builds a representation of the system M , usually as a graph-based data structure called Binary Decision Diagrams (BDDs), which can be used to represent boolean functions. States represent possible configurations, while transitions are changes from one configuration to another. Probabilities are assigned to the transitions between states, representing rates of negative exponential distributions.

Let $\mathbb{R}_{\geq 0}$ be the set of positive reals and AP be a fixed, finite set of atomic propositions used to label states with properties of interest. A labeled CTMC \mathcal{C} is a tuple $(S, \bar{s}, \mathbf{R}, L)$ where:

- S is a finite set of states;
- $\bar{s} \in S$ is the initial state;
- $\mathbf{R} : S \times S \rightarrow \mathbb{R}_{\geq 0}$ is the transition rate matrix, which assigns rates between each pair of states;
- $L : S \rightarrow 2^{AP}$ is a labeling function which labels each state $s \in S$ the set $L(s)$ of atomic propositions that are true in the state.

The probability of a transition between states s and s' being triggered within t time-units is $1 - e^{-\mathbf{R}(s,s') \cdot t}$. The elapsed time in state s , before a transition occurs, is exponentially distributed with the *exit rate* given by $E(s) = \sum_{s' \in S} R(s, s')$. The probability of changing from state s to s' is given by $P(s, s') = \frac{\mathbf{R}(s,s')}{E(s)}$ [Parker, 2002]. Finally, the probability of changing from s to s' in t time units is given by [Clarke et al., 2008]:

$$P(s, s', t) = \frac{\mathbf{R}(s, s')}{E(s)} \times (1 - e^{-\mathbf{R}(s,s') \cdot t})$$

A computational path of a CTMC model starting at state s_0 is an infinite sequence $\pi = s_0 t_0 s_1 t_1 \dots$, where $s_i \in S$, $t_i \in \mathbb{R}_{\geq 0}$ is the time spent at state s_i and $R(s_i, s_{i+1}) > 0$ for all $i \geq 0$. The Figure 4.11 shows the graph of the CTMC model below:

- $S = \{S_0, S_1, S_2\}$;
- $S_0 = \{S_0\}$;
- $R(S_0, S_1) = \lambda_1$, $R(S_1, S_2) = \lambda_2$, $R(S_2, S_0) = \lambda_3$ and $R(S_1, S_0) = \lambda_4$;

- $L(S_0) = \{A, B\}$, $L(S_1) = \{A, \neg B\}$ and $L(S_2) = \{\neg A, B\}$.

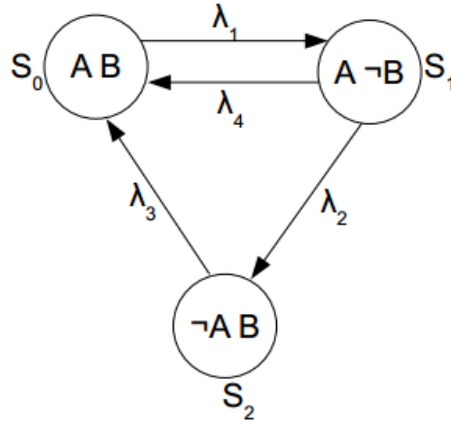


Figure 4.11: An example of a CTMC model.

4.3.1 Probabilistic Logics

Properties can be expressed quantitatively as “What is the probability of overtaking-vehicles without car-crash?” or qualitatively as “Is it safe to do the overtake maneuver?”, offering valuable insight over the system behavior.

Properties are specified using the Continuous Stochastic Logic (CSL) [Kwiatkowska et al., 2008], which is based on the Computational Tree Logic (CTL) and the Probabilistic Computation Tree Logic (PCTL). The syntax of CSL formulas is the following:

$$\Phi ::= true \mid a \mid \neg\Phi \mid \Phi \wedge \Phi \mid \mathcal{P}_{\triangleleft p}[\phi] \mid \mathcal{S}_{\triangleleft p}[\phi]$$

$$\phi ::= \mathbf{X} \Phi \mid \Phi \mathbf{U}^I \Phi$$

where a is an atomic proposition, $\triangleleft \in \{>, <, \geq, \leq\}$, $p \in [0, 1]$ and I is an interval of $\mathbb{R}_{\geq 0}$.

There are two types of CSL properties: transient ($\mathcal{P}_{\triangleleft p}$) and steady-state ($\mathcal{S}_{\triangleleft p}$). In this work we are interested in transient or time related properties. A formula $\mathcal{P}_{\triangleleft p}[\phi]$ states that the probability of the formula ϕ being satisfied from a state respects the bound $\triangleleft p$. Path formulas use the \mathbf{X} (next) and the \mathbf{U}^I (time-bounded until) operators. For example, formula $\mathbf{X} \Phi$ is true if Φ is satisfied in the next state.

This can be applied to check if one state leads to another with a probability p , for example, state “open-in” is followed by state “open-out” with at least 10% chance: $\mathcal{P}_{\geq 0.1}[\text{“open-in”} \implies \mathbf{X} \text{“open-out”}]$.

4.3.2 PRISM

There are several model checkers for stochastic processes. Each tool presents its own features, some more complete than others, a renowned tool is the PRISM. It supports different types of models, properties and simulators [Kwiatkowska et al., 2011]. It has been largely used in distinct fields, e.g. communication and media protocols, security and power management systems. We have used PRISM in this work for several reasons, which include: exact PMC in order to obtain accurate results; Continuous-time Markov Chain (CTMC) models, suited for our field of study; rich modeling language that allowed us to build our model; and finally property specification using Continuous Stochastic Logic (CSL), which is able to express qualitative and quantitative properties.

4.3.2.1 Modeling

One way to model the vehicular movement in the PRISM language is presented in the Figure 4.12. We have modeled two vehicles moving in the same direction however in different lanes on a freeway. A description of a CTMC model in PRISM must begin with the keyword `ctmc`. It is composed of modules, which have their states represented by a set of variables which assume a finite set of values. In the example, there are two modules for each vehicle: `Mod_vX`, `Mod_dX` to represent respectively, speed and position. In speed modules there is a variable which describes the current velocity, which varies from 0 to the constant `desired_speed_car`. The other modules there is a variable which describes the current position, which varies from 0 to the constant `RS`. The new values to speed and position are calculated respectively by formulas: $v = v_i + at$ and $x = x_i + vt + (0.5)at^2$.

Typically, a variable declaration specifies the initial value for that variable. The initial state for the model is then defined by the initial value for all variables. It is possible, however, to specify that a model has multiple initial states. This is done using the `init...endinit` construct. Between the `init` and `endinit` keywords, there should be a predicate over all the variables of the model. Any state which satisfies this predicate is an initial state. In our example, we have just set the initial position up of *vehicle 1*. Thus, all states which have the value 1 to the variable `pos_c1` will be initial states.

The behavior of the modules is defined by the transitions between states. These transitions are defined by commands expressed as `[action] g → r : u`. This command indicates that if the predicate `g` (also known as conditions or guard) is observed (true), then the system will be updated by `u`, which is composed of one or more declarations expressed as $x' = \dots$, indicating that the value of x is updated (x' is the value of

```

ctmc

const time = 1; //seconds
const int RS = 300; //road size (meters)
const double a = 3.0; //m/s^2
const int desired_speed_car = 23; //m/s
formula mov_c2 = v_c2+(a*pow(time,2))/2;
formula mov_c1 = v_c1+(a*pow(time,2))/2;

init pos_c1=1 endinit

// --VEHICLE 1--
module Mod_vC1 //speed
v_c1 : [0..desired_speed_car];

[m] (v_c1 <= desired_speed_car) ->
    v_c1' = min(max(ceil(v_c1+a*time),0),
                desired_speed_car);
endmodule

module Mod_dC1 //position
pos_c1 : [1..RS];

[m] (pos_c1 <= RS) ->
    pos_c1' = min(
        (ceil(pos_c1 + mov_c1)),RS);
endmodule

// --VEHICLE 2--
module Mod_vC2 //speed
v_c2 : [0..desired_speed_car];

[m] (v_c2 <= desired_speed_car) ->
    v_c2' = min(max((ceil(v_c2 + a*time)),0),
                desired_speed_car);
endmodule

module Mod_dC2 //position
pos_c2 : [1..RS];

[m] (pos_c2 <= RS) ->
    pos_c2' = min( (ceil(pos_c2 + mov_c2)),RS));
endmodule

rewards ''speed''
    true : v_c2;
endrewards

rewards ''time''
    true : 1;
endrewards

```

Figure 4.12: PRISM model of the vehicular movement.

variable x in the next state). The value of the rate at which the update will occur is defined by r .

PRISM also allows the synchronization between modules, through labels which must be in brackets at the start of the synchronized commands. Transitions in different modules using the same label happen simultaneously. The resulting rate is equal to the product of the individual command rates of each synchronized module. In the example, the movement between vehicles 1 and 2 (given by the kinematics Equations) is represented by the commands labeled with m . The values assignment rate is omitted because they assume the default value 1. In the other words, it is certain that both vehicles will move in each step of the model. Thus, the final rate when this movement occurs is $1 \times 1 \times 1 \times 1$ and it is given by the product of the rates of the four commands labeled with m .

4.3.2.2 Rewards

PRISM also allows including **rewards** in the model, which are structures used to quantify states and transitions by associating real values to them. The state rewards are counted proportionately to the elapsed time in the state, while transition rewards are counted each time the transition occurs. In PRISM, rewards are described using

```

rewards "horizontal"
  (semaphore = false) : 1;
endrewards
```

Figure 4.13: An example of a state reward.

the following syntax:

```

rewards "reward name"
...
endrewards
```

Each reward is specified using multiple reward commands which follow the syntax below.

$$[sync] guard : reward;$$

Reward commands describe state rewards and transition rewards, respectively. The predicate which must be observed is the *guard*. The *sync* is a label used to synchronize a set of commands into a single transition in the system. Finally, *reward* is an expression, which can contain variables and constants from the model, and when evaluated it counts for the reward.

One example of a state reward is the traffic light signal (**semaphore**, in our model), described in Figure 4.13. The *reward* is one, since it is essentially counting the number of times when that particular reward was observed. The *guard* is the condition which must be observed – **semaphore=false**, that state must be present. The *sync* is absent here, because this event does not need to be synchronized with another module. Therefore, the cumulative reward represents how many times the traffic light was open to the horizontal lane on a crossroad (Value *true* represents the vertical one).

Reward properties can be applied to states and transitions. For example, “What is the expected reward for the semaphore open to the vertical lane on a crossroad at time T?”.

This reward can be instantaneous, obtaining its value at the given time through the property $\mathcal{R}_{=?}[\mathcal{I}^=t]$, or accumulated, calculating its value until the given time, using the property $\mathcal{R}_{=?}[\mathcal{C}^{<=t}]$. In this work, we have used both rewards because they can show the history, accumulating values such as the time of an event and instantaneous ones to get information as current speed or acceleration.

Rewards of paths in a Continuous-time Markov chain are summations of state

rewards along the path and transition rewards for each transition between these states. State rewards are interpreted as the rate at which rewards are accumulated, essentially counting them, i.e. if t time units are spent in a state with state-reward r , the accumulated reward in that state is $r \times t$.

4.3.2.3 Property Specification

The properties to check CTMC models in PRISM must be specified in the Continuous Stochastic Logic (CSL), which is a temporal logic based on Computation Tree Logic (CTL), Probabilistic CTL (PCTL) and reward-based extensions [Kwiatkowska et al., 2008]. The CSL formulas use the following syntax:

$$\begin{aligned} \Phi & ::= \text{true} \mid a \mid \neg\Phi \mid \Phi \wedge \Phi \mid \mathcal{P}_{\leq p}[\phi] \mid \mathcal{S}_{\leq p}[\phi] \\ \phi & ::= \mathbf{X} \Phi \mid \Phi \mathbf{U}^I \Phi \end{aligned}$$

where a is an atomic proposition, $p \in [0, 1]$ is a probability and I is an interval of $\mathbb{R} \geq 0$ (real non-negative numbers) at which the property must be met. The operators \neg and \wedge are logical ones, while \mathbf{X} and \mathbf{U} are temporal operators. The symbol $\leq \in \{>, <, \geq, \leq\}$ represents the type of bound which the property must satisfy. For example, if \leq is $>$, the probability of the property must be higher than p .

There are two basic types of CSL properties: transient ($\mathcal{P}_{\leq p}$) and steady-state ($\mathcal{S}_{\leq p}$). This work focused on transient and reward-based properties, therefore we will describe the semantics of only transient ones.

The formula $\mathcal{P}_{\leq p}[\phi]$ is true in state \mathbf{s} if the probability that ϕ is satisfied by a path starting at state \mathbf{s} matches the bound $\leq p$. Path formulas are built using the operators \mathbf{X} (*next*) and \mathbf{U}^I (*time-bounded until*). The path formula $\mathbf{X} \Phi$ is true if Φ is satisfied in the next state, while $\Phi^1 \mathbf{U}^I \Phi^2$ is true if Φ^2 is satisfied at some time unit in the interval I and in all previous time units Φ^1 is satisfied.

Other operators can be created from this minimum set of CSL operators, such as the \mathcal{G} (always) operator. The interval I can be omitted from the operators \mathbf{U} and \mathbf{F} , which means that $I = [0, \infty)$. Finally, one can also quantify the probability of a property ϕ by using the expression $=?$ instead of the bound $\leq p$ ($\mathcal{P}_{=?}[\phi]$).

A few examples of transient properties are presented below considering the PRISM model previously discussed (Figure 4.12).

- $\mathcal{P}_{=?} [\mathbf{F}^{[0,T]} \text{ pos_c1 } \geq \text{RS}]$: the probability of the `vehicle 1` across the road at `T` time units. This is a $\mathbf{F}^I \Phi$ (eventually) property, where Φ is the atomic proposition `pos_c1 >= RS`;

- $\mathcal{P}_{=?} [(\text{pos_c1} < \text{pos_c2}) \mathbf{U} (\text{pos_c2} \geq \text{RS})]$: the probability of the `vehicle 2` arrives before `vehicle 1`. This is a $\Phi_1 \mathbf{U}^I \Phi_2$ (time-bounded until) property, where $I = [0, \infty)$ and Φ_1 and Φ_2 are the atomic propositions `pos_c1 < pos_c2` and `pos_c2 >= RS`, respectively.

Furthermore, PRISM also allows to check the expected value of model rewards. Some of the properties of this type which will be used throughout this work have the forms $\mathcal{R}_{\triangleleft r} [I = t]$, $\mathcal{R}_{\triangleleft r} [\mathbf{F} \Phi]$ and $\mathcal{R}_{\triangleleft r} [\mathbf{C} \leq t]$, with r and $t \in \mathbb{R}_{\geq 0}$.

The first property ($\mathcal{R}_{\triangleleft r} [I = t]$) is true, starting from a state s , if the state reward at the instant t satisfies the bound $\triangleleft r$.

The second property ($\mathcal{R}_{\triangleleft r} [\mathbf{F} \Phi]$) is true, starting from s , if the accumulated reward along the path until the point where Φ is true satisfies the limit $\triangleleft r$.

Finally, the third property ($\mathcal{R}_{\triangleleft r} [\mathbf{C} \leq t]$) is true, starting at s , if the accumulated reward along the path at instant t satisfies the bound $\triangleleft r$.

Given the definition of a reward, its accumulated value along the path in a CTMC model is the sum of the state rewards along the path, plus the sum of the transition rewards between these states, both defined in the body of the same reward structure. The state reward associated with each state is $v \times t$, where t is the time spent at the state and v is the state reward associated with the state. If the bound is not specified, using the expression $=?$ one obtains the expected value of that reward.

A few examples of properties which obtain the reward values for the considered example are presented below, given the rewards defined in the Figure 4.12.

- $\mathcal{R}\{\text{"speed"}\}_{=?} [\mathbf{I} = T]$: the instantaneous velocity of the `vehicle 2` after T time units;
- $\mathcal{R}\{\text{"time"}\}_{=?} [\mathbf{F} (\text{pos_c1} \geq \text{RS})]$: the expected time for the `vehicle 1` to across the scenario.

4.3.2.4 Filters

PRISM will by default return the value for the initial states of the model. However, since model checking is exhaustive and computes exact answers, values are usually generated for all states of a model. For example, when model checking $\mathcal{P}_{=?}[\mathbf{F} \text{fail}]$, PRISM computes the probability of reaching a state in which `fail` is true, starting from any state of the model [Kwiatkowska et al., 2009]. Therefore, it is possible customize PRISM properties to obtain others results according to different ranges of states. This is done using filters, which are created using the **filter** keyword. They take the following form:

filter(*op, prop, states*)

where *op* is the filter operator, *prop* is any PRISM property and *states* is a Boolean-valued expression identifying a set of states over which to apply the filter.

Here's a simple example of a filter: **filter**(max, $\mathcal{P}_{=?} [(\text{pos_c1} < \text{pos_c2}) \text{ U } (\text{pos_c2} \geq \text{RS})], \text{pos_c2}=\text{RS}/2$ **and** $\text{pos_c1}=0$) This gives the maximum value, starting from any state satisfying $(\text{pos_c2}=\text{RS}/2)$ **and** $(\text{pos_c1}=0)$, of the probability of the `vehicle 2` arrives before `vehicle 1`.

Most filters of the form `filter(op, prop, states)` apply some operator “op” to the values of property “prop” for all the states satisfying `states`, resulting in a single value. The full list of filter operators in this category is:

- *min*: the minimum value of prop over states satisfying states
- *max*: the maximum value of prop over states satisfying states
- *count*: counts the number of states satisfying states for which prop is true
- *sum* (or *+*): sums the value of prop for states satisfying states
- *avg*: the average value of prop over states satisfying states
- *first*: the value of prop for the first (lowest-indexed) state satisfying states
- *range*: the range (interval) of values of prop over states satisfying states
- *forall* (or *&*): returns true if prop is true for all states satisfying states
- *exists* (or *|*): returns true if prop is true for some states satisfying states
- *state*: returns the value for the single state satisfying states (if there is more than one, this is an error)

There are also a few filters that, rather than returning a single value, return different values for each state, like a normal PRISM property:

- *argmin*: returns true for the states satisfying states that yield the minimum value of prop
- *argmax*: returns true for the states satisfying states that yield the maximum value of prop

- *print*: does not change the result of prop but prints the (non-zero) values to the log
- *printall*: like print, but displays all values, even for states where the value is zero

4.3.2.5 PMC Implementation

The techniques which are implemented in PRISM to check the properties of CTMC models with rewards include graph theory algorithms and numerical computation. The first ones are used on the graph structure which represents the Markov chain specified in the tool to determine, for example, the set of reachable states or to check qualitative properties. In this case, the algorithms are executed on the BDDs as it happens on the non-probabilistic version of the model checking technique.

The numerical computation is required to solve a Markov chain and calculate the probabilities and reward values (quantitative properties). Iterative methods such as Jacobi and Gauss-Seidel are used to solve systems of linear equations and check properties such as $\mathcal{S}_{\leq p}$, $\mathcal{P}_{\leq p} [\Phi_1 \mathbf{U} \Phi_2]$ and $\mathcal{R}_{\leq r} [\mathbf{F} \Phi]$. The iterative method known as uniformisation is used to calculate rewards and probabilities for properties which involve a time interval \mathbf{I} or a specific time \mathbf{t} : $\mathcal{P}_{\leq p} [\Phi_1 \mathbf{U}^t \Phi_2]$, $\mathcal{R}_{\leq r} [\mathbf{I} = \mathbf{t}]$ and $\mathcal{R}_{\leq r} [\mathbf{C} \leq \mathbf{t}]$. Further details on these techniques to solve Markov chains can be found in [Parker, 2002].

Furthermore, to determine the quantitative properties using numerical computation, PRISM allows the use of three data representations:

1. a generalization of BDDs, known as Multi-terminal BDDs (MTBDDs) to represent real-valued functions, given that matrices and real vectors are required. These data structures allows compact representations and efficient manipulation of big models because they explore their regularities. However, the computation is often slow;
2. explicit representation, as a sparse matrix, which allows a faster and direct computation, however, it can not deal with big models;
3. a hybrid approach, which extends the MTBDDs, allowing faster computations than compared to their original representation. In this case, MTBDDs are used to represent only the transition matrix, while the solution vectors of the iterative methods are represented by traditional real-valued vectors.

The adjacency matrix below illustrates the explicit representation of the transitions of the CTMC model of the Figure 4.15a.

$$\begin{pmatrix} 2 & 5 & - & - \\ 2 & 5 & - & 7 \\ - & - & - & - \\ - & 7 & - & - \end{pmatrix}$$

Figure 4.14: The adjacency matrix that explicitly represents the CTMC model.

Consider that the first line and first column of the matrix are 0-indexed. The entry $M(l, c)$ of the matrix indicates the value of the line l and column c . For example, the value of the entry $M(0, 0)$ is 2, while the value of $M(1, 3)$ is 7. The Figure 4.15b shows an example of a MTBDD which represents the transitions of the CTMC model of the Figure 4.15a.

In the representation through MTBDD, the binary variables x_1 e x_2 encode the indexes of the transition matrix lines, while y_1 e y_2 encode the indexes of the columns. For example, for the entry $M(1, 3)$ of the transition matrix, the index 1 of the line is encoded through the binary representation $(x_1, x_2) = (0, 1)$ and the index 3 of the column is encoded as $(y_1, y_2) = (1, 1)$. Thus, by following the path $x_1 = 0, y_1 = 1, x_2 = 1$ and $y_2 = 1$ in the MTBDD (note that the order of the variables is $x_1 < y_1 < x_2 < y_2$) the terminal node 7 is reached, which is the value of the transition matrix for the entry $M(1, 3)$.

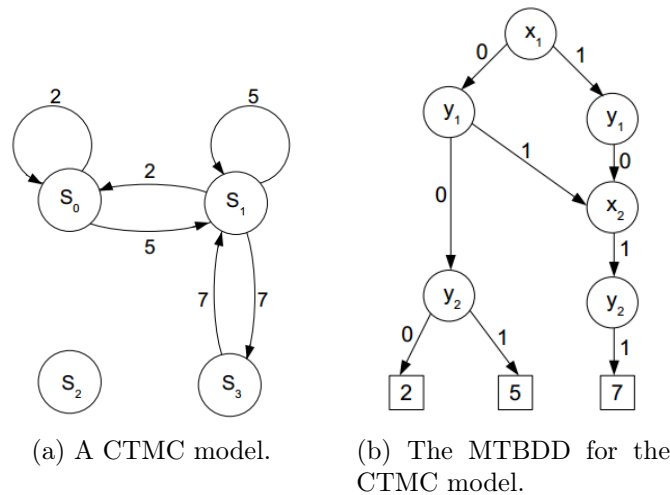


Figure 4.15: A CTMC and its MTBDD representation

Further details on the techniques used by the PRISM tool can be found in [Kwiatkowska et al., 2011, 2007, 2004]. In the remainder of this chapter we present the related works to this thesis. Three case studies of model checking are discussed, depicting how the academic community has been modeling Wireless Sensor Networks

(WSNs) and VANETs. The first study focuses on the verification of a traffic light synchronization protocol. The second study proposes a graphic-based specification of wireless networks protocols, it can be used as a modeling environment for both simulation and model checking. The third case provides an analysis of probabilistic model checking applied to a specific congestion control protocol of VANETs.

4.4 Applying Model Checking to Wireless Sensor Networks and Vehicular Networks

Verification of distributed systems such as wireless sensor networks (WSNs) is difficult to be performed. Even a single sensor node can adopt position or unexpected behavior into environment, therefore, tests are a highly non-trivial task. Besides, these systems can be used in areas where they can not be reprogrammed. Thus, it is recommended to verify that each node, and the network as a whole, fulfill their requirements, otherwise implementation failures can be costly and may cause accidents. Traditional approaches such as simulators do not verify the desired properties for all possible computations of a distributed system. It is known that software defects often appear only in specific situations. Furthermore, such bugs can not be reliably detected by simulation approaches.

A formal verification approach which can be used in such complex systems is model checking [Christian, 2009; Cimatti et al., 2000]. Several researchers have used model checking to test systems requirements and protocols. In Naik and Sistla [1994], the authors have verified the IEEE 802.3 Ethernet CSMA/CD protocol using the symbolic model checker SMV [McMillan, 1992]. Fehnker et al. [2007] have checked the LMAC protocol, a medium access control protocol for WSNs using the Uppaal model checker [Berhmann et al., 2006]. Currently, there are several studies in the literature about analysis of VANET protocols, however, heretofore few works have been using the model checking approach [Konur and Fisher, 2011].

Next sections explore some of the most relevant works. In Christian [2009], the authors present the use of model checking to analyze WSNs. They check a four-way traffic light crossroad using WSNs. The system follows a protocol to manage a synchronization signal in each lane. The authors have used the NuSMV [Cimatti et al., 2000] tool to identify potential failures in the protocol. In another study, Fehnker et al. [2009] propose a graphical tool for the specification of wireless networks which in turn generate models for the Castalia simulator and the PRISM probabilistic model checker (PMC). The wireless connection behavior is based on an analytical model which has

been validated empirically [Zuniga and Krishnamachari, 2004]. Finally, in Konur and Fisher [2011], the authors present one of the few studies which uses probabilistic model checking in VANETs. The authors have verified the correctness of a network congestion Control Protocol.

4.5 Reliable Model Checking for WSNs

Christian [2009] describes the verification of a traffic light synchronization protocol on a four-way intersection. However, they have proposed the verification of WSNs in general. The authors outline guidelines and abstractions to improve the verifiability of WSNs. Their objective was to make model checking easier to apply in this field, which helps to achieve faster and correct verification of WSN models and make formal analysis amenable in the WSN domain.

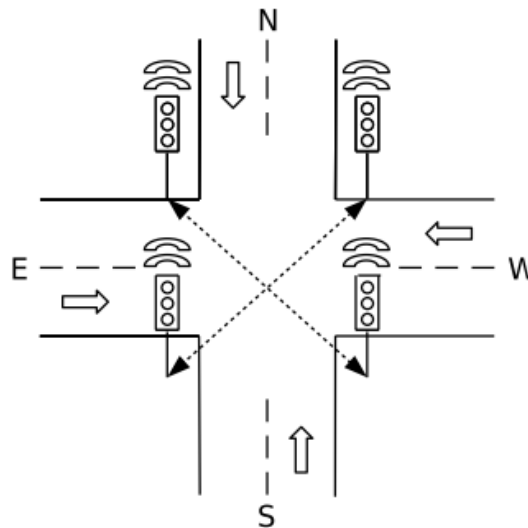


Figure 4.16: A schematic of the four-way crossroad intersection with traffic lights which have been modeled using probabilistic model checking. Adapted from [Christian, 2009]

The purpose of the analyzed protocol is to synchronize traffic lights through wireless connections and to ensure that only diagonally aligned traffic lights are allowed to show the green signal at the same time (see Figure 4.16 for a schematic representation).

The main contribution of this paper was the survey on the communication modeling pitfalls and how to properly solve them. The author has shown that the nonobservance of package collisions and the impossibility to listen to the channel while communicating might cause the deadlock of traffic signals. During verification, without

considering such aspects, researchers could not find a counter-example for the property that states that “only diagonally aligned traffic lights are allowed to show the green signal at the same time”. However, when variations of radio wave propagation were introduced in the model, the verification results showed computation paths where three traffic lights could show the green signal at the same time, which could potentially lead to traffic accidents.

Therefore, Christian [2009] suggests to model the wireless communication channel using DEFINE statements, which work similar to *macros*. Statements for the free channel, with packet collision and others situations of traffic lights have been specified. The value of these DEFINE statements are determined by the current control states of the traffic lights and the value of input variables. The control states *sendReq*, *sendAck*, *sendAckPartner* and *sendComplete* are used in the verification model. The first control state is used to represent that a message was sent to inform the other traffic lights of a request. The control state *sendAck* is used to check that the traffic light received a request. The control state *sendAckPartner* is used when the traffic light has received a request from the diagonally aligned traffic light at the intersection. The last control state (*sendComplete*) notifies the other traffic lights of a successful traffic light change.

Figure 4.17 shows examples of DEFINE commands for following situations: 1) the channel being free and 2) message collisions. For the first command, the DEFINE holds the boolean value *true* if no other traffic light at the intersection currently sends a message, otherwise *false*. In order to detect message conflicts, the DEFINE named “collision” assumes the value *true* if two or three other traffic lights simultaneously send messages [Christian, 2009].

Define statements

```

DEFINE
free:= !(sendReqEast|sendAckEast|sendPartnerEast|sendCompleteEast|...);
collision:= !((sendReqEast|sendAckEast|sendPartnerEast|sendCompleteEast) &
              (sendReqWest|sendAckWest|sendPartnerWest|sendCompleteWest)...);

```

Figure 4.17: DEFINE commands for the communication channel modeling. One command represents that the channel is free and can be used, and the other one represents that a message collision has happened. Adapted from [Christian, 2009].

The author has concluded that system components such as synchronization protocols often can not be verified isolated in WSNS. A formal model of the communication protocol should observe others sensor node components, such as timers or even parts of operating systems. They indicate that one challenge is to find suitable models which do not affect the verifiability of the system and correctly describe

its intended behavior. Therefore, specially for non-verification experts, suitable and faultless abstraction techniques should be available [Christian, 2009].

4.6 Graphical modelling for simulation and formal analysis of wireless network protocols

Fehnker et al. [2009] proposed a graphical-style of specification that acts as a modelling environment for simulation and model checking. This tool is called CaVi. It provides an unified modelling interface to the Castalia simulator [Boulis, 2013], and model checking tool PRISM [Kwiatkowska et al., 2011].

Castalia is a Wireless Sensor Network simulator which it is able to test distributed algorithms and protocols within a realistic wireless channel and radio model which takes into account the physical characteristics of the radio. Figure 4.18 shows the proposed architecture. The gossip and flooding protocols are supported. In this case, nodes listen to a message and then forward it; once a node has received and sent a message, the node becomes inactive.

CaVi generates PRISM models automatically using generic templates based on flooding and gossiping protocols. The novelty is that the behavior of the wireless links is based on an empirically validated model proposed by Zuniga and Krishnamachari [2004], the same model used in Castalia. This signal propagation model is presented in Section 3.4. The generated PRISM models can be either synchronous or asynchronous. The synchronous models assume that nodes receive or send at the same time, while the asynchronous models allows for an arbitrary delay between reception and transmission. The latter covers all possible interleavings; something that is difficult if not impossible to achieve by simulation [Fehnker et al., 2009].

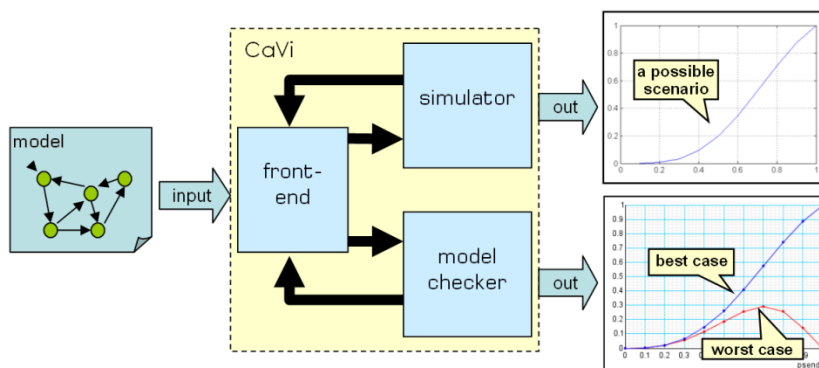


Figure 4.18: CaVi Architecture – Font [Fehnker et al., 2009]

According to Fehnker et al. [2009], the main feature of CaVi is its graphical interface, with which a user can design a specific network layout. Nodes may be created in a "drag-and-drop" fashion, and the properties of individual nodes (such as the power and signal strength) may be tuned as necessary via individual node menus of parameters.

4.6.1 Analytic model to PRISM

Figure 4.19 shows a small network example which will be translated to the Prism model taken from Fehnker et al. [2009]. Four nodes are numbered from 0 to 3. The first and the last one correspond to the Source and Target node respectively.

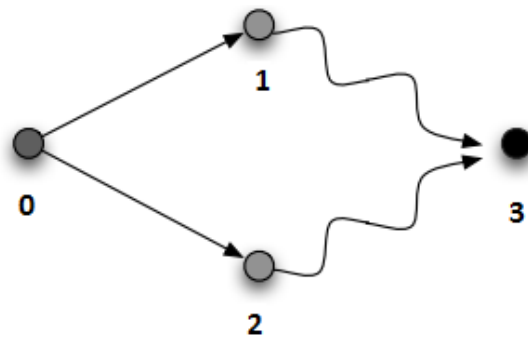


Figure 4.19: Network example – Adapted by [Fehnker et al., 2009]

All nodes are either active or inactive; when a node “*i*” is active ($active_i = 1$) it can listen for and/or receive a message, or send one it received previously. The authors use the variable *send_i* to denote whether node “*i*” is in possession of an uncorrupted message ($send_i = 1$) which it must forward, or not ($send_i = 0$ – maybe because one was never received, or because it has already been forwarded). Figure 4.20 formalizes this behavior for node 3, where *recvp₃* is the link probability which depends on the state of the surrounding nodes, and whose calculations are shown in the Section 3.4.

Synchronous behaviour for node

```

module node3
active3:[0..1] init 1;
send3: [0..1] init 0;

[tick] send3=0&active3=1 -> recvp3:(send3'=1)&(active3'=1)+
(1-recvp3):(send3'=0)&(active3'=1);
[tick] send3=1&active3=1 -> send3'=0&active3'=0;
[tick] active3=0 -> send3'=0&active3'=0;
endmodule

```

Figure 4.20: PRISM code for node 3

In this synchronous style all nodes behave in lockstep, synchronized on the action *tick*. The difference in behavior whether one node or several nodes broadcast at the same time is all accounted for in the link probabilities. The authors computed the various quantities such as the transmission powers, the signal-to-noise ratios and thresholds for each node, taking into account their actual pairwise separations. CaVi precomputed $rx_{i,j}$ from Equation 3.11, the power at the receiver “i” from message broadcast by “j”. These values are denoted in the PRISM model by `linRxSignal_i_j`.

```
const double linRxSignal_1_3 = 3.04330129123453E-8;
const double linRxSignal_3_1 = 3.04330129123453E-8;
```

Figure 4.21: $rx_{i,j}$ constants

Signal-to-noise ratio $SNR(i, j)$ between pairs of nodes calculated from equations given by 3.11, 3.12 and 3.13 are depicted in Figure 4.22.

```
formula snr_3_1 = (linRxSignal_3_1*send3)/
  (linRxSignal_0_1*send0 + linRxSignal_2_1*send2 + 1.0E-10);
```

Figure 4.22: Examples of the SNR from node 3 to 1.

Next the conditional link probabilities $precv_{i,j}$ from Equation 3.16 are calculated from the precomputed thresholds, and combined in a single PRISM formula, with $precv_{3,2}$ given as an example (Figure 4.23),

```
formula Preceive_3_2 = func(max,0,(snr_3_2>=12.357925578002547)?
  func(pow,(1-0.5*func(pow,2.71828,-0.781*snr_3_2)),8*25)
  :0);
```

Figure 4.23: Examples of the $precv$ from node 2 to 3.

Total link probabilities P_i are computed as the sum, following Equation 3.17.

```
formula recvp0 = func(min,1,Preceive_1_0+Preceive_2_0+Preceive_3_0);
formula recvp1 = func(min,1,Preceive_0_1+Preceive_2_1+Preceive_3_1);
formula recvp2 = func(min,1,Preceive_0_2+Preceive_1_2+Preceive_3_2);
formula recvp3 = func(min,1,Preceive_0_3+Preceive_1_3+Preceive_2_3);
```

Figure 4.24: Probabilities “p” of connection.

Finally, verifications can be presented. For example, Figure 4.25 computes the separated probabilities that nodes 1, 2 and 3 obtain the message. The probability can vary according to position of each node.

Through model, the researchers confirmed that using a simple flooding protocol, the communication suffers from some serious performance issues. Since, the nodes send

Some temporal logic properties

(1) $P_{min}=?$ [send1 = 0 U send1 = 1]

What is the minimum probability of the active node 1 waiting for communication until it obtains the message.

(2) $P_{min}=?$ [send2 = 0 U send2 = 1]

What is the minimum probability of the active node 2 waiting for communication until it obtains the message.

(2) $P_{min}=?$ [send3 = 0 U send3 = 1]

What is the minimum probability of the active node 3 waiting for communication until it obtains the message.

Figure 4.25: Checking basic properties

as soon as they receive the message, the transmission is affected by high likelihood of interference. To mitigate this problem, variations of this basic protocol were verified in which nodes only forward a received message with probability p . Thus, the model helps to choose the value of p to optimize the probability of the message being received by all nodes [Fehnker et al., 2009].

4.7 Formal Analysis of a VANET Congestion Control Protocol through Probabilistic Verification

Konur and Fisher [2011] provide an analysis through the use of probabilistic model checking to a specific congestion control protocol for VANETs. Probabilistic model checker PRISM was used to investigate the correctness and effectiveness. A congestion control protocol is an algorithm which is used to share available resources among nodes within a network [Wei et al., 2006]. It is used when available resources are limited.

Vehicular Networks present peculiar features and the traditional congestion control protocol does not guarantee reliable and safe communication. Thus, studies have been proposing new protocols and they need to be tested. According to Konur and Fisher [2011], in most of these, however, the analysis of a proposed method relies on simulations for an evaluation of its efficacy. Yet, such simulations can examine only a limited subset of all possible behaviors.

The analyzed protocol was proposed for Bouassida and Shawky [2010] and it is based on “dynamic scheduling” and “transmission of priority-based messages”.

Thus, priorities are assigned to messages dynamically, and high-priority messages are transmitted in preference to low-priority ones. In order to provide a reliable and safe network it is important to ensure fast delivery of emergency messages without any delay. Thus, this protocol has three stages [Konur and Fisher, 2011]:

- **Dynamic priority assignment** A priority is assigned to a message based on the utility of the message. The priorities determine when the messages are transmitted.
- **Message scheduling** Based on the assigned priorities, messages are sent to an appropriate channel. In a VANET, packets are accessed through a shared medium [Torrent-Moreno et al., 2005]. For example, control channel (CCH) and service channels (SCHs).
- **Cooperative message transmission** For high priority messages, Bouassida and Shawky [2010] adopt the following message transmission process: “the transmission of low priority messages is frozen, even if their corresponding channel is free.” Whenever a message is sent from a channel, the package with the highest priority within the channel is selected.

After choosing the congestion protocol, Konur and Fisher [2011] elected key characteristics for their model:

1. The number of vehicles within an interference range at any time can be any value in $\{0, \dots, n\}$
2. There are three types of messages: (i) emergency messages driven by an event – with generation rate from $\{0, \dots, m\}$; (ii) periodic safety messages; and (iii) periodic non-safety service messages. Generation rates for safety and non-safety messages are constant (represented by k and l , respectively).
3. There are a CCH and SCH channels. The available bandwidth for each channel is assumed to be same. Half of the available bandwidth for CCH is devoted to the transmission of emergency messages; the other half is used for safety messages.
4. We also assume that, if the number of messages in CCH and SCH exceeds a certain threshold value, the channel is overloaded and the excess messages are lost
5. The counting abstraction approach was adopted, whose overall behavior can be captured.

Figure 4.26 shows a state machine for emergency messages. The diagrams for other two types of messages are similar. The difference appears if the service channel is overloaded and the safety channel is empty. Thus, any excess service messages are forwarded to the control channel. However, if there are messages in safety channel, then the service messages are lost.

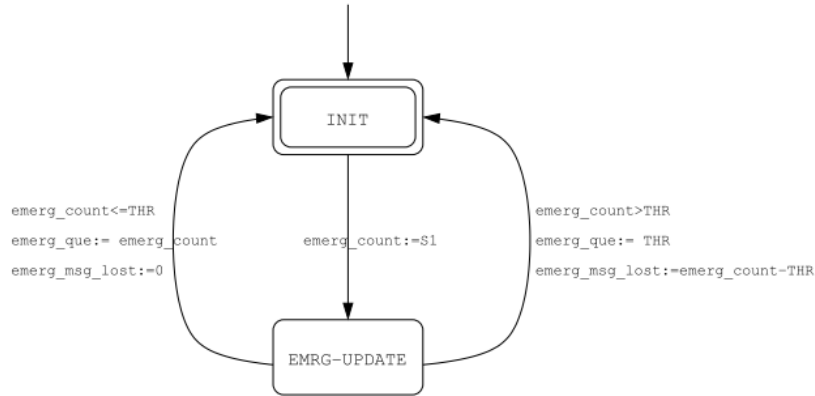


Figure 4.26: Transition system for emergency message control. – Font [Konur and Fisher, 2011]

This state machine updates the emergency message queue and counts the number of lost messages. The variable $S1$ denotes the new size queue which it is calculated by Equation 4.1. At each time instant new emergency messages (denoted as $emerg_msg$) are added to the queue (denoted as $emerg_que$) and $MSG_TRNS/2$ messages are transmitted from the queue. If the queue size is smaller than the threshold ($emerg_count \leq THR$), no message is lost ($emerg_msg_lost := 0$). Otherwise excess messages are lost ($emerg_msg_lost := emerg_count - THR$).

$$S1 = emerg_que + emerg_msg - \min(emerg_que, MSG_TRNS/2) \quad (4.1)$$

Finally, some properties were explored. Based on the proposed model, the correctness and performance was evaluated with respect to selected PCTL properties. Figure 4.27 shows some of them. PRISM returned *true* for properties I and II, showing that the congestion control protocol correctly works for emergency messages. However, the third property returned *false*. It shows that the observed delays in safety messages sometimes are higher than the delays for service messages.

Through several observations, Konur and Fisher [2011] proposed some modifications like: (a) periodic safety messages use all available bandwidth within the control channel, when an emergency message arrives, it takes precedence and is

transmitted without any delay; (b) service messages only use the service channel, if this channel is overloaded, the control channel is not used for service messages. Nevertheless, final analysis showed slightly increased over that of the original method. The modifications proposed allowed to achieve a better (i.e. lower) loss rate and delay for safety messages.

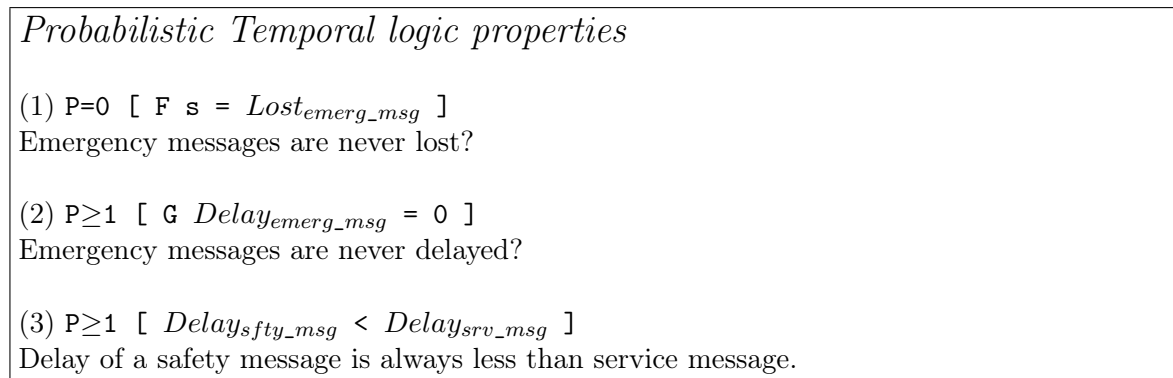


Figure 4.27: Checking basic properties – Adapted by [Konur and Fisher, 2011]

4.8 Conclusion

In this chapter we have presented model checking, a technique used to model and analyze systems, verifying if those respect a set of properties. The symbolic model checking is presented, as well as its concepts and related subjects, such as the model representation using the Kripke structure and the data structure binary decision diagram. The properties to be verified are specified using special types of logic, the temporal logics, Computational Tree Logic (CTL and CTL*) and Linear Time Logic (LTL), which are also described. Using these logics, it is possible to specify different types of properties, such as “if a message is sent, then that message is eventually received”, or “a distributed system never enters a deadlock state”.

The probabilistic model checking is also presented, an extension to the previous technique which allows the modeling and analysis of stochastic systems. Different structures are necessary and were described, such as multi-terminal binary decision diagrams, and probabilistic logics, Probabilistic Computational Tree Logic (PCTL), Continuous Stochastic Logic (CSL), reward- based and filter extensions. PRISM model checkers was examined and their modeling language and property specification were briefly presented.

The case studies depicted here show that verification techniques can be very useful in assessing the efficiency and correctness of WSN and VANET, thus, their

results may be used to improve the systems. Presented studies show how to model the communication with different abstractions and model checkers.

Nevertheless, researches in VANET rarely use formal methods in their analysis. Bouassida and Shawky [2010] and Lomuscio et al. [2010] are exceptions. They employ model checking, however, the studies do not address uncertainty and non-determinism. Another important point to mention is the absence of models which represent the dynamism of the nodes movement and the non-determinism caused by vehicles in traffic. These issues are determinants in vehicular networks. Table 4.1 shows a comparison among the presented works here.

Table 4.1: Model Checking Research

Research	Model Checker	Network Type	Analysis Type	Movement	Signal propagation type
Christian [2009]	NuSMV	WSN	Correctness	-	deterministic
Fehnker et al. [2009]	PRISM	WSN	effectiveness	-	stochastic
Konur and Fisher [2011]	PRISM	VANET	Correctness/ effectiveness	-	deterministic

The focus of all works mentioned above was to verify the communication protocol. However, it is important verifying networks considering not only communication by itself, but also implemented functionality. Thus, it is often necessary to model communication and other important system components to verify functionality [Christian, 2009]. Therefore, building models considering traffic flow, network computers and radio-propagation is necessary and rarely explored.

Chapter 5

Our Models to Verify Movements in VANETs

Outline. In this chapter we present our proposal for the formal verification of vehicular movement using two models using probabilistic model checking. First, a macroscopic vision about a crossroad managed by a semaphore with different parameters is modeled. Then, an overtaking vehicle scenario is encoded. It uses analytical formulas to represent position, speed and acceleration providing microscopic aspects.

5.1 Macroscopic model

Our experimental model represents different VTL protocols. Signal propagation and communication have been abstracted assuming an Isolated approach. This first model shows a wider vision, i.e., it is a macroscopic model because the vehicles are represented by integer values. Thus, dimensions, position, acceleration and velocity are neglected.

Classifying our model according to the features presented in Section 3.1, it does not include the movement from origin to destination, its vehicles randomly choose their direction (Trip Modeling level). We modeled a single intersection and therefore it was not necessary to define routes (Path Modeling level). In the Flow Modeling level, we have modeled a VTL managed crossroad, where each road has two lanes in opposite directions. This has been done to demonstrate the viability of PMC usage to check which factors influence mobility patterns, such as, number of cars, road size and semaphore timer duration.

Figure 5.1 illustrates transit dynamics in our model. Vehicles in vertical lanes (`top1` and `bot2`) await the vertical semaphore, while vehicles in horizontal lanes are free to move. Vehicles have the option of following in the lane direction or turn

right. Figure 5.1 also shows the standard adopted to feed back the cars on crossroad (Continuous Traffic Transitions label). After leaving `left1`, a vehicle is reinserted in `right1` - an analogous behavior is used in vertical lanes. This solution was adopted in order to maintain generality of the model while the modeling includes only one crossroad.

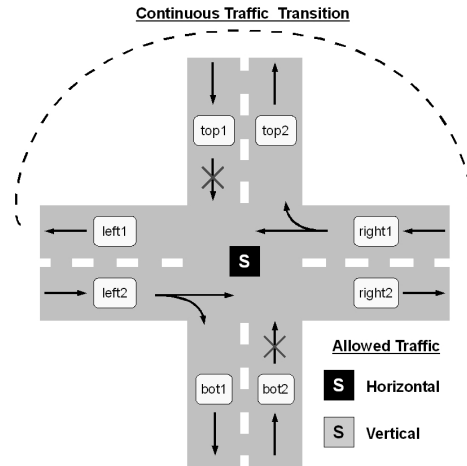


Figure 5.1: Mobility VANET Model

Our model follows the characteristics of the VTL protocol presented in Section 3.7.4, with the following abstraction — although a leader is temporarily elected and responsible for sending messages to open or close the roads, these messages and the leader have been simplified into a global variable per crossroad. This is due to the objective of verifying the vehicles movement instead of communication protocols and signal propagation.

The VTL Protocol model is written in the PRISM language (used by the PRISM model checker) and consists of a module that represents the roads managed by the virtual traffic light. A fragment of the model is shown in Figure 5.2 and its complete version can be found in the supplementary material at [Ferreira et al., 2012a].

The `semaphore` variable indicates which road is open for traffic. Each lane of the roads is represented by an integer number which stores the vehicles in that lane for a time frame (e.g., `lane_right1`). Vehicle movement is represented by modifying one unit of the variables which represent the lanes of origin and destination, respectively.

Model behavior is parametrized by the constant `protocol`, which defines one of three possible modes. In the first one, the horizontal road has precedence over the vertical road and stays open for a fixed-timer period, in other words, a classic traffic light (`protocol=0`). The second mode favors the most intensive road, this is a new version proposed by the authors of this work, because it does not use timer,

```

VTL Protocol PRISM Model

const int MAX_TIMER=15;
const int protocol; // 0 - timed signal; 1 - Most intensive traffic VTL

module crossroad
  timer : [1..MAX_TIMER] init 1;
  // Vehicles in both lanes
  // First method - timed signal with initial preference to horizontal lane
  [] (protocol=0) & (lane_right1>0 | lane_left2>0) & (lane_top1>0 | lane_bottom2>0) &
    (timer = MAX_TIMER) -> 1 : (semaphore'!=(semaphore)) & (timer'=1);

  // Second method - Chooses the most intensive lane
  [] (protocol=1) & (lane_right1>0 | lane_left2>0) & (lane_top1>0 | lane_bottom2>0) &
    ((lane_right1+lane_left2)>=(lane_top1+lane_bottom2)) &
    (timer = MAX_TIMER)-> 1 : (semaphore'=false);
endmodule

```

Figure 5.2: VTL Protocol PRISM Model

i.e., the traffic flow is analyzed all the time and the semaphore can change anytime ($protocol=1$). The third protocol also verifies the most intensive traffic, however it obeys a timer when a lane is chosen – a classic virtual traffic light ($protocol=2$). In these latter two options, when there is traffic in only one road, the semaphore opens for this road – this prevents vehicles remain stopped unnecessarily.

The experimental model presented in Figure 5.2 is a macroscopic view, where the vehicles obey certain traffic rules and choose a random direction. However, probabilities can easily be assigned to indicate a tendency of direction or PoI. Moreover, a microscopic vision can be obtained by increasing the granularity of the model by using modules to represent vehicles. Thus, the modeling can be done hierarchically in different levels, as shown in Section 3.1, which are the trip, path and flow modeling.

5.2 Results for the macroscopic model

There are two variables in our model which can be parametrized: initial number of cars per lane and the maximum value for the semaphore timer. The first variable sets the initial value for each side of each lane, which behaves as a queue and is limited by the constant `MAX_CARS`. If we set this initial value to zero or `MAX_CARS`, the model becomes trivial since there is nothing happening — either due to no cars in any lane or because it is too crowded and cars can not move. On the other hand, intermediary values ($\lceil \frac{MAX_CARS}{2} \rceil$) significantly complicate the model, since there are many possible transfers between lanes. The behavior of the model Statistics (size), as well as the time to build and check it has been shown in Table 5.1, where we have varied the number of cars from zero to `MAX_CARS`.

Table 5.1: Model statistics for different cars per lane (*timer* = 15).

Cars per Lane	States	Transitions	T_{Build}	$T_{semaphore}$	$T_{traffic}$
0	15	15	0.032 s	0.0010 s	0.0020 s
1	97260	553886	1.222 s	0.023 s	4.462 s
2	1341720	8973247	26.597 s	0.164 s	148.829 s
3	1192440	7692693	11.43 s	0.281 s	126.925 s
4	72600	361920	0.864 s	0.015 s	5.062 s
5	30	30	0.033 s	0.0020 s	0.0030 s

The second variable sets the maximum value for the semaphore timer, which is used to guarantee that a road has a sufficient amount of time when it is given the green light. Model complexity is linear in function of the semaphore timer (Table 5.2), where we have varied the semaphore timer from 9 to 21.

Table 5.2: Model statistics for different semaphore timers (*cars per lane* = 2).

Timer	States	Transitions	T_{Build}	$T_{Semaphore}$	$T_{Traffic}$
9	805032	5380177	10.322 s	0.172 s	88.825 s
12	1073376	7176712	12.532 s	0.19 s	118.903 s
15	1341720	8973247	10.436 s	0.163 s	148.831 s
18	1610064	10769782	11.955 s	0.197 s	178.336 s
21	1878408	12566317	10.246 s	0.214 s	207.48 s

The model also allows changing the VTL protocol employed (either favoring the vertical lane or the most intensive lane) through the flag `protocol`. The columns T_{Build} , $T_{Semaphore}$ and $T_{Traffic}$ refer to the time to build the model, and to check a semaphore safety property and a traffic reward property. The experiments have been performed in an Intel(R) Xeon(R) CPU X3323, 2.50GHz which has 16 GB of RAM memory.

5.2.1 Semaphore, Lane and Timer Safety Properties

We have formulated some properties related to the behavior of different model aspects in order to verify if it shows the expected behavior, such as semaphore interchange and timer reset. These properties can be seen in Figure 5.3. The first property checks if the semaphore interchanges appropriately between one value (`false` — horizontal) and the other (`true` — vertical). In order to express this we have used the operators $P \geq 1$ for certainty (therefore we expect the result `true`), G for every starting path with semaphore set to horizontal, implication (\Rightarrow) to indicate causality between the events and F for a path with semaphore eventually set to vertical.

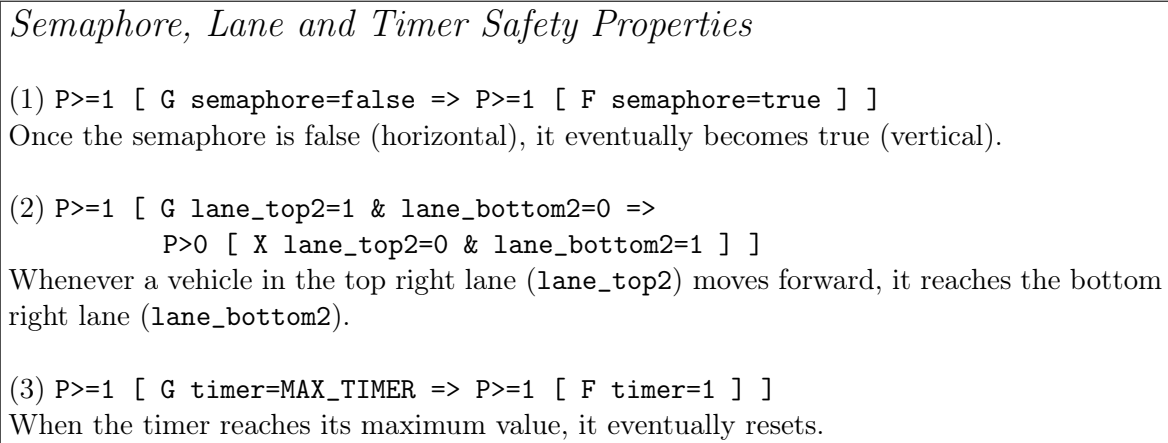


Figure 5.3: Semaphore, Lane and Timer Safety Properties

We also have checked if the continuous traffic transitions behave as expected, which means that if a vehicle moves to outside of top lane (`lane_top2`), it should appear in the bottom right lane (`lane_bottom2`). Finally, we have checked if the timer reaches its maximum, it eventually resets. All results were correct.

5.2.2 Semaphore and Lane Quantitative Properties

In order to quantify the behavior of the semaphore and the lanes over time, we have added to the model several rewards and some examples are described in Figure 5.4. These rewards essentially count each time some given condition is *true*. The semaphore rewards are “horizontal” and “vertical”, which are true when the variable `semaphore` is `false` and `true`, respectively. One of the lane rewards is `lane_right1`, which is always true and counts its number of vehicles.

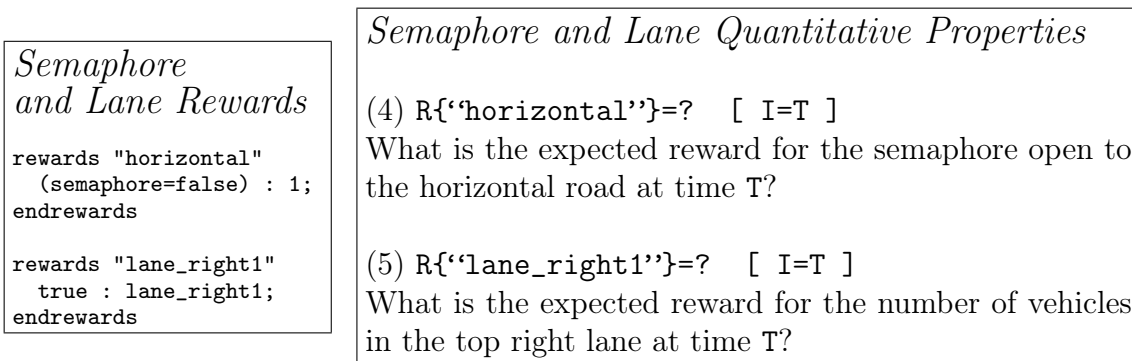


Figure 5.4: Semaphore and Lane Rewards and Quantitative Properties

Since we have included these rewards in our model, we are able to quantify them over time, using the reward and instant operators. The I (instant) operator checks a property at a precise given time T. The R (reward) operator is used to query the value

of some given reward, for example, $R\{\text{‘lane_right1’}\}=?$. Therefore, the properties shown in Figure 5.4 query the value of the rewards at a precise given time T .

5.2.3 Traffic Comparison of VTL and Timed Traffic Lights

The vehicular traffic in a horizontal and vertical road were analyzed for a crossroad managed by a common signal (*timed*) and two instances of VTL with different modes - (1) VTL which obeys a time (*classic VTL*) and (2) new VTL which does not have timer (*intensive*). Figure 5.5 shows that the timed signal oscillates between clustering and absent of traffic. For the second instance of VTL, there is a continuous flow without clustering of vehicles and a reduction of time on lane. The first option of the VTL shows an average behavior between the two methods mentioned. Furthermore, the graph shows an abrupt oscillation in the beginning. This occurs due a design option, the horizontal lane always starts open for traffic, thus, the reduction of vehicles on own lanes is certain.

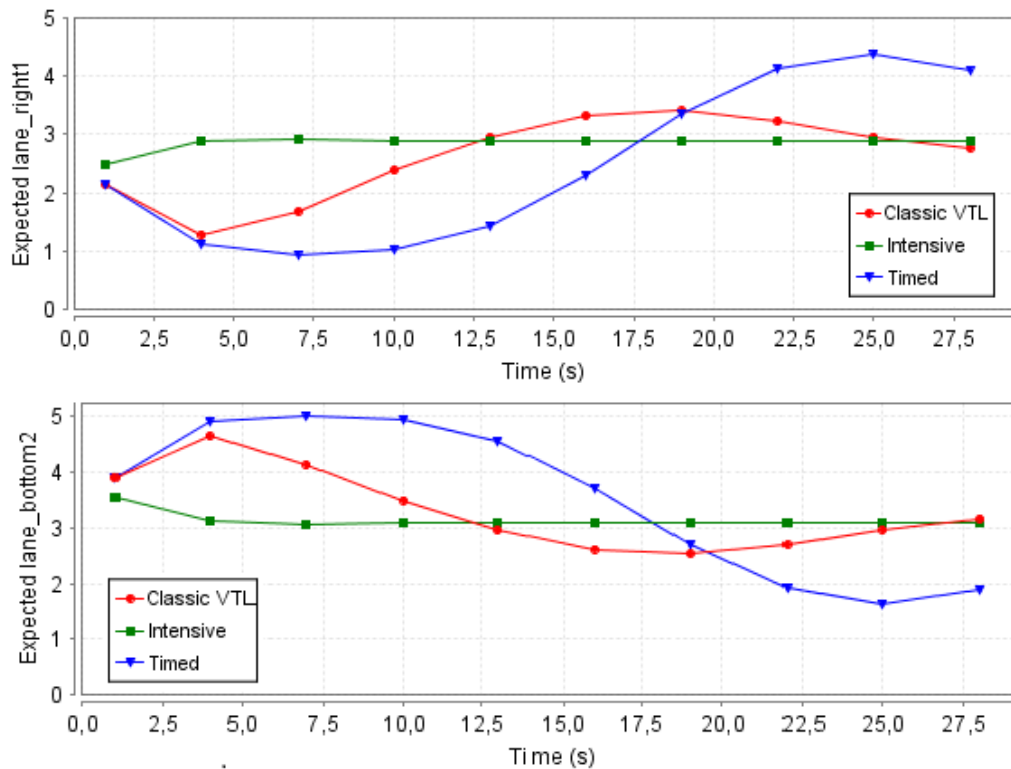


Figure 5.5: Traffic Comparison of VTL and Timed Traffic Lights

The mirrored behavior of the timed signal was expected. However, the behavior of the two VTL instances shows unexpected features, such as, the average number of cars on crossroad equals for both directions, which indicates that these behaviors could

be more efficient and fair in practice, because it avoids the agglomeration of vehicles by interleaving the active road with more intensive traffic.

5.2.4 The Influence of Traffic and Timer in Semaphore Interchange

Semaphore Interchange Property

(6) P=? [semaphore=false U<=T semaphore=true]

What is the probability that the semaphore changes from **false** (horizontal) to **true** (vertical) within T seconds?

Figure 5.6: Semaphore Interchange Property

The traffic volume and upper limit or the timer influence directly the probability of the semaphore changing from horizontal to vertical. We have devised a property shown in Figure 5.6 which queries the probability of the semaphore changing from **false** (horizontal) to **true** (vertical) in T seconds.

We have plotted this probability in Figure 5.7 for the three protocols with different number of cars. As time progresses, the probability rises, which means it is likely that the semaphore changes from horizontal to vertical. However, the Intensive and Timed protocols are insensitive to traffic increase. The first one changes the direction in a few seconds due traffic jam in opposite lane, while the timed protocol is stable until reaches *MAX_TIMER*. In the classic VTL protocol, the timer and the number of vehicles must be queried, thus, the probability increases slower than the Intensive protocol, however, it is still better than the Timed protocol.

5.3 Microscopic model

Our second model was created with a microscopic focus. The idea is to show the representation of nodes movement through the analytical equations presented in Section 3.2. Signal propagation and communication have been abstracted assuming an Isolated approach. Our microscopic model takes into account position, speed, and acceleration of the vehicles. For this, an overtaking vehicle scenario is implemented. This has been done to demonstrate the viability of PMC usage to check microscopic aspects. This model also is written in the PRISM language. Fragments of the models are presented below and the complete version can be found in the supplementary material at [Ferreira et al., 2014a].

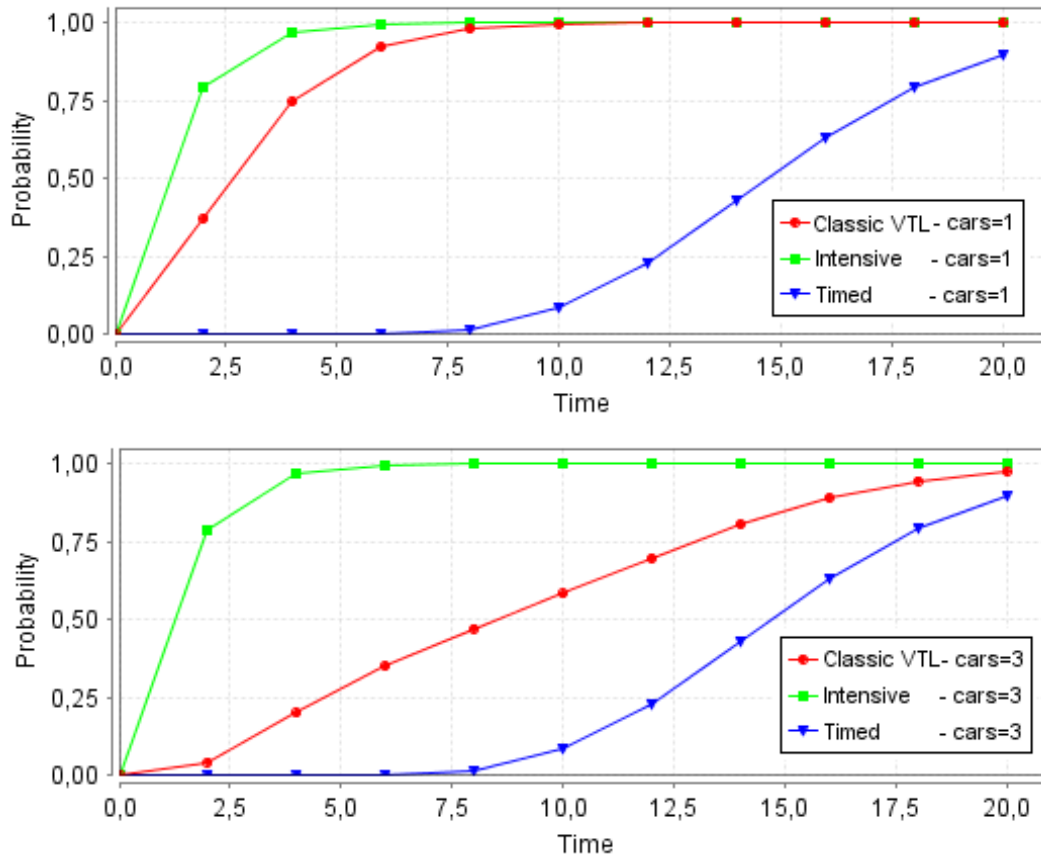


Figure 5.7: Probability of Semaphore Interchange for each Protocol

Figure 5.8 illustrates the proposed scenario. There are three vehicles involved. The car $c1$ will overtake the truck, called *Leader*, which travels slower. However, the vehicle $c2$ is coming in the opposite direction. In this situation, $c1$ may not see $c2$, due to weather conditions or lack of attention. This scenario will happen in 200 meters. Thus, the model should answer questions such as “What is the probability of a collision?”.

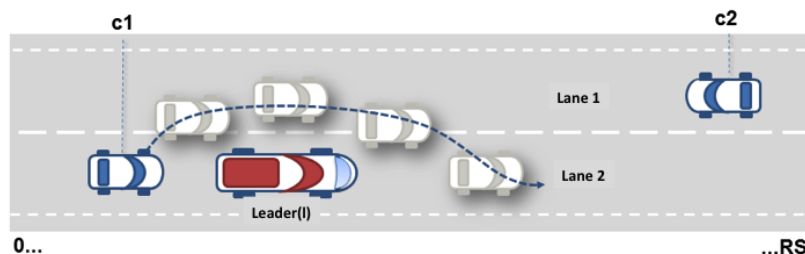


Figure 5.8: Overtaking vehicle scenario

Figure 5.9 depicts the models variables. Each vehicle maintains its current position and velocity. The variable *lane* informs where $c1$ is located. If *lane* is equal to 1, the vehicle is on right-hand side (default road), otherwise the car is on

left-hand side. In other words, the vehicle is trying to overtake. The constants `desired_speed_car`, `desired_speed_truck` and `RS` (Road side) constrain the model and they are respectively represented in m/s , m/s and m . The `carCrash` variable indicates whether $c1$ and $c2$ collided at some point in time.

```

Model variables

lane      : [1..2]; //lane's c1 (1 - right lane, 2 - left lane)
carCrash  : bool;

// speed
v_c1 : [0..desired_speed_car];
v_c2 : [0..desired_speed_car];
v_l  : [0..desired_speed_truck]; //leader speed

// position
pos_c1 : [1..RS];
pos_c2 : [1..RS];
pos_l  : [1..RS]; //initial value should obey a minimum distance (truck_size+min_gap_car)

```

Figure 5.9: Variables of model

This model becomes interesting because it does not have a specific initial state. This is achieved by the code shown in Figure 5.10. The implemented predicate states that vehicles $c1$ and $c2$ in the opposite directions are separated by `RS` meters and there is a leader (truck) between them, which will be overtaken by $c1$. However, the leader position and the initial speed of all involved can be a combination of values. This creates several scenarios to be automatically explored. An interesting abstraction was adopted to $c2$'s position. It starts at position *one* of its lane, however, its real location on the road is given by $RS - pos_c2$. We have chosen to insert the car $c1$ in the left lane ($lane = 2$) if it is forced to slow down in the initial state, $c1$ also starts at this lane whether its initial speed will overlap the truck's position.

```

Initialization of variables

init
  (pos_l>=truck_size+min_gap_car)&(pos_c1=1)&(pos_c2=1)&
  (lane=( (v_c1>pos_l)|(a_c1 <= 0)?2:1))&
  (v_c1>=0&v_c1<=desired_speed_car)&
  (v_c2>=0&v_c2<=desired_speed_car)&
  (v_l>=0&v_l<=desired_speed_truck)&
  (carCrash=false)
endinit

```

Figure 5.10: Initial states for the model

The vehicles position is given by Equation 3.3 implemented in the PRISM language, which involves the initial position, velocity, acceleration, and time. Each transition of the model represents a time period that is defined by the constant t . The

accelerations of the vehicles are calculated by the IDM model previously presented in Section 3.2. The new speed is given by Equation 3.4 and it also depends on the vehicle acceleration. Figure 5.11 describes a fragment of the model responsible for calculating the acceleration and position of vehicle $c1$. The formulas are similar for other vehicles.

As mentioned in Section 3.2, the IDM expression combines the free-road acceleration strategy, given by $a_{free}(v) = a[1 - (v/v_0)^\delta]$ with a deceleration strategy, given by $a_{brake}(s, v, \Delta v) = -a(s^*/s)^2$. Therefore, Equation 3.1 has been algebraically split during implementation, because the vehicles do not suffer deceleration when there are no obstacles ahead. Thus, when the vehicle $c1$ overtakes the *leader*, $c1$ does not suffer slowdown, while the truck's acceleration, which used to have free way, starts to be influenced by the new $c1$'s position.

Acceleration and Position Formulas

```

formula a_c1_free = AM_car - AM_car * pow(v_c1 / desired_speed_car, exponent);
formula a_c1_obst = a_c1_free - a_brake_c1;
formula a_c1 = (overtook|lane=2?a_c1_free:(pos_l>=RS?a_c1_free:a_c1_obst));

formula a_brake_c1 = AM_car * pow(des_dyn_dis_c1 / deltaD_c1, 2);
formula des_dyn_dis_c1 = min_gap_car + max(0.0, v_c1 * T_car + (v_c1 * deltaV_c1) /
                                           (2*pow(AM_car*BM_car,0.5) ));

formula deltaV_c1 = v_c1 - v_l;
formula deltaD_c1 = max(pos_l - pos_c1 - truck_size,1);/"max 1" to avoid division by zero

formula mov_c1 = (v_c1 + ( a_c1*pow(time,2)) / 2) > 0 ?
                 (v_c1 + (a_c1*pow(time,2)) / 2) : (-1 * (v_c1 + (a_c1*pow(time,2)) / 2));

```

Figure 5.11: IDM model implementation

The *Mod_vC1* module and the *Mod_dC1* presented in Figure 5.12 are responsible for the transitions in the model assigning the new position and speed to vehicle $c1$, again, the modules are similar to other vehicles involved. The *ChangingLane* module controls the lane change of $c1$. In case the vehicle is able to overtake according to the conditions presented by Mobil model (Subsection 3.2.1), the vehicle changes to left lane. If $c1$ is on the left lane and already overtook the leader, then $c1$ returns to the default lane. This modules are synchronized by label “m” what it is placed inside the square brackets. This last module is also responsible for detecting a crash, which happens when $c1$ and $c2$ are in the same lane and the coordinates are overlaid or the deceleration calculated by the Torricelle equation (Equation 3.5) is unfeasible to be executed in a normal situation.

```

Modules proposed

module Mod_vC1
  // speed
  [m] (pos_c1 <= RS)&(v_c1 <= desired_speed_car) ->
      (v_c1' = min(max(ceil(v_c1 + a_c1)*time,0),desired_speed_car));
endmodule

module Mod_dC1
  // position
  [m] (pos_c1 <= RS) -> (pos_c1' = min(ceil(pos_c1 + muv_c1),RS));
endmodule

module ChangingLane
  [m] (pos_c1 <= RS) -> (lane' = ((lane = 2)&(pos_c1 >= (pos_l+min_gap_car+car_size)))?1:
      ((lane = 1)&(can_change_lane))?2:lane);

  [] (CanotDeceleration | (
      ( (pos_c1>=(RS-pos_c2))&(pos_c1<=(RS-pos_c2-car_size)) )
      |
      ( ((pos_c1-car_size)>=(RS-pos_c2))&((pos_c1-car_size)<=(RS-pos_c2-car_size)) )
      )
      ) & (lane=2) -> (carCrash' = true);
endmodule

```

Figure 5.12: Modules implementation

5.4 Results for the microscopic model

Finally, the model built using PRISM language can be verified. The idea is to check the correctness of IDM code and analyzing different situations about the modeled scenario. The behavior of the model, as well as the time to build and check are shown in Table 5.3, where we have 7 properties. For some, we have varied the number of initial states by command *filter*. The experiments have been performed in an Intel(R) Xeon(R) CPU X3323, 2.50 GHz which has 17 GB of RAM memory.

Table 5.3: Model statistics for different proprieties.

Figure – Property	States	Transitions	Initial States	$T_{Building}$	T_{Check}
Figure 5.13 – 1	386243	386243	38400	2384.512 s	1.91 s
Figure 5.15 – 1	386243	386243	38400	2115.274 s	2.656 s
Figure 5.15 – 4	386243	386243	38400	2119.281 s	0.333 s
Figure 5.16 – 5	386243	386243	1	2111.286 s	11.739 s
Figure 5.16 – 6	386243	386243	1	2118.044 s	11.943 s
Figure 5.19 – 8	386243	386243	38400	2107.819 s	5.418 s
Figure 5.19 – 9	386243	386243	38400	2360.838 s	2.876 s

First, we have formulated properties related to the correctness of different aspects of the model in order to verify if it shows the expected behavior, such as the speed or acceleration. For example, the vehicles never exceed the desired limit of the road. In order to express this we have used the operators $P \geq 1$ for certainty (therefore we expect

the result `true`), \mathbf{G} to demonstrate for every path the velocity is not greater than the allowed speed along the road. This property can be seen in Figure 5.13. The second property checks if the vehicles will be in motion as soon as the model is initiated. Finally, we have checked if the velocity never reaches negative values in some situation as deceleration movements. For three properties the results were *true*, which they were the expected values. The verifications were made for vehicle *c1*, however they can be easily modified to others involved.

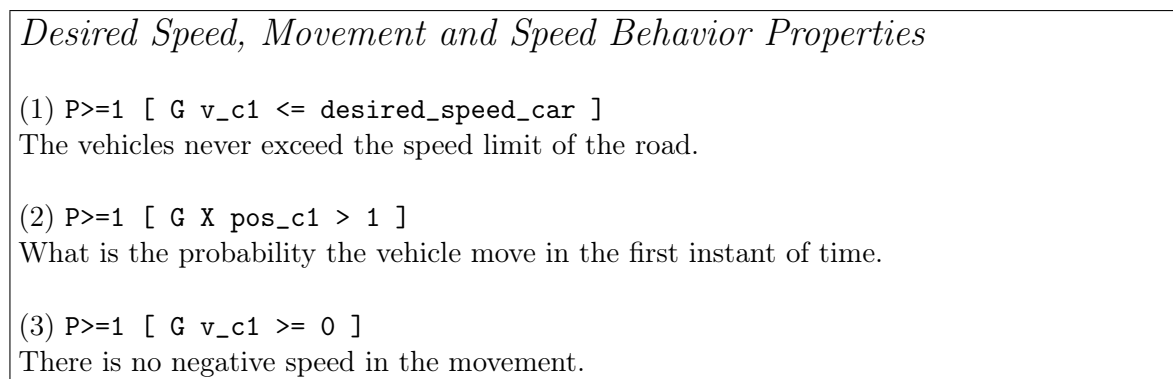


Figure 5.13: Correctness properties

5.4.1 Car-crash situation

In order to analyze some situations about the scenario, several interesting questions can be made. For example, the first property in Figure 5.15 checks the probability of occurrence of car-crash. The presented result was: $[0.0, 1.0]$ for range of values over initial states. The answer shows that there are situations without accident, however car-crash also happen to vehicles. The third property in the same figure checks the average probability of accident taking into account all initial states. Thus, this scenario has 98 percent of initial states resulting in collision. The fourth property only confirms the results of these two properties mentioned before. The high accident rate happens because *c1* always try the maneuver independently of the initial states. Thus, the vehicle *c1* executes the overtaking even starting with zero meters by second, an unreal conditional in practice. It is a non-probabilistic query and the result was *true* for the question “Is there any situations without accident?”. The \mathbf{E} (there exists) operator asks whether some path from a state satisfy a particular path formula. If the result is true, a witness will be generated. In this case, the model checker returned a counterexample depicted in Figure 5.14, which represents the initial state with values to the respective variables.

Step		Mod_vC1	Mod_vC2	Mod_vL	Mod_dc1			Mod_dc2	Mod_dL
Action	#	v_c1	v_c2	v_l	pos_c1	lane	carCrash	pos_c2	pos_l
	0	0	0	0	1	1	false	1	19

Figure 5.14: Counterexample to situations without accidents

Coming backing to the second property, it answers the question, “is there a possibility to finish the scenario without overtake?”, in other words, the *leader* reaches the end before *c1*. The result was $[0.0, 1.0]$ considering the range of values over initial states. Thus, there are cases with overtake and others without this movement.

Car-crash Scenario Properties

(1) P=? [F (carCrash=true)]

What is the probability of an accident occurs?

(2) P=? [((pos_c1<RS & carCrash=false)U(pos_l>=RS & carCrash=false))]

What is the probability of not occurring overtakes in this scenario?

(3) filter(avg, P=? [F carCrash=true], "init")

What is the average probability over all initial states that an accident occurs?

(4) E [F (carCrash=false) & (pos_c1>=RS)]

Is there at least one path which does not lead to an accident?

Figure 5.15: Properties of overtake

As we mentioned above, the operator **E** has an excellent feature to generate a counterexample (a path reaching a “goal” state). Using this witness, for instance to property 4, we can thoroughly analyze the situation of accidents in the scenario. Since we have included rewards in our model, we are able to quantify the speed, acceleration and movement over time in this counterexample. Some implemented rewards and properties are shown in Figure 5.16. The latter using the *filter* command to check specifically the counterexample available. The operator **R** is responsible for getting the values from rewards commands.

Figure 5.17 depicts the result of analysis. The graphs show the position of the three vehicles over the time. The line with triangular point-shape (label *lane for car1*) varies between 10 and 20 and it shows the lane of the vehicle *c1* during overtaking. The first value means that *c1* is in the default lane (right-hand lane), the value 20 means that the vehicle is traveling in the left-hand lane to overtake. The Graph 5.17a shows the behavior of vehicles without collision.

Note that *c1* overcomes the leader at the instant 7.5 and when the positions of *c1* and *c2* overlap, the first one already returned to the right-hand lane. Similar to

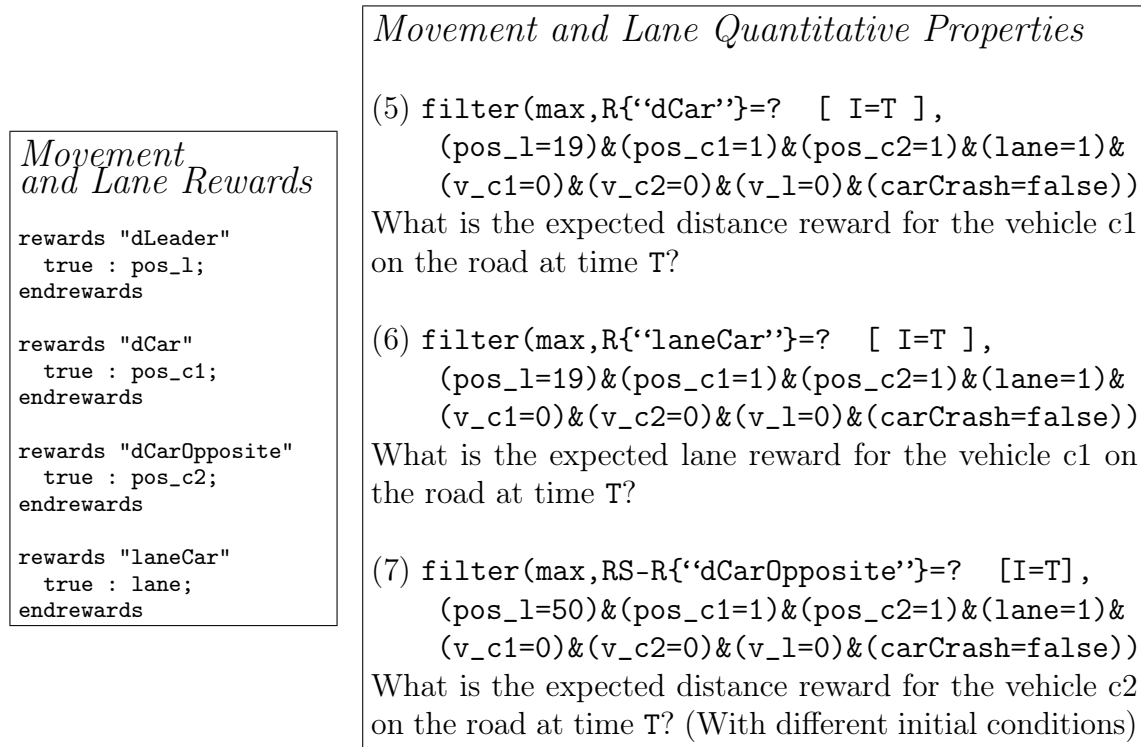
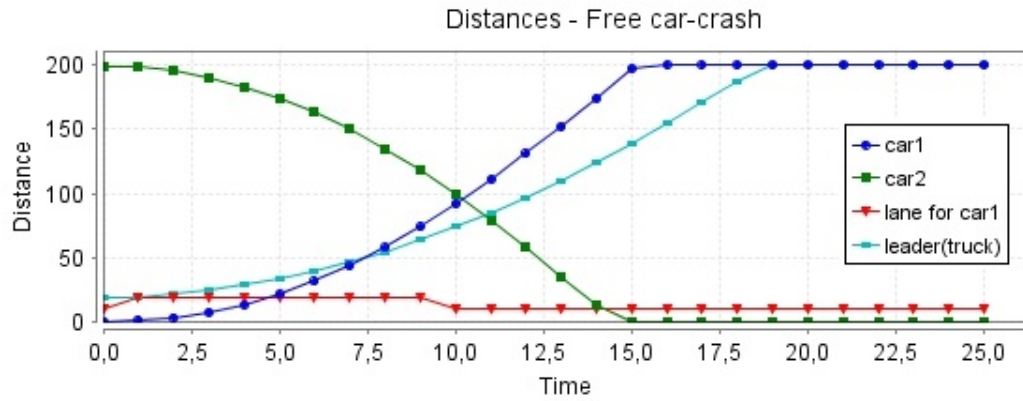


Figure 5.16: Movement and Lane Rewards and Quantitative Properties

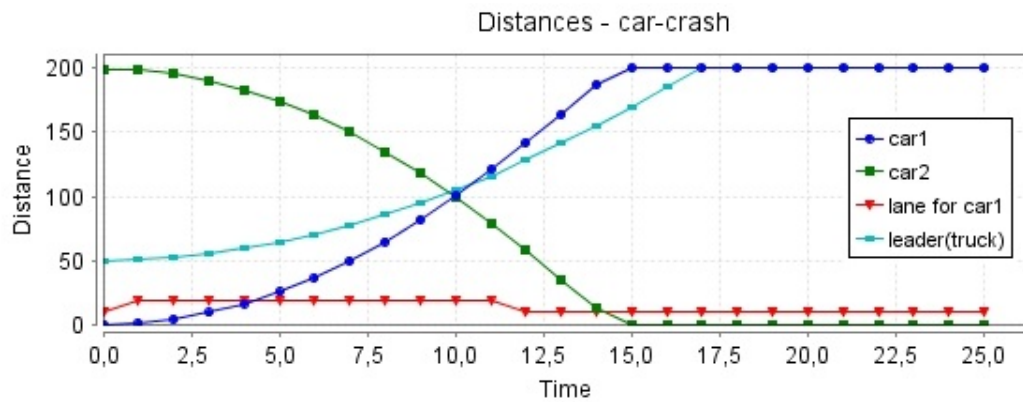
property 4, we can generate counterexamples of an accident scenario. The Graph 5.17b shows this situation and it is easy to notice the positions overlapped at time 10. In this case, *c1* is in the left-hand lane indicating the collision.

Figure 5.18 shows the evolution of the acceleration and velocity of *c1* and *leader* (truck) in the scenario of overtaking without collision. Speeds rise according to acceleration until reaching the maximum limit of the road. As the acceleration of the truck is slower and the speed limit is lower, the car can overtake with ease. It is interesting to note that the acceleration modeled with IDM is affected by lane change of *c1*. Right at the instant 2, the acceleration of *c1* rises abruptly. Because in this moment, the driver concludes to be more advantage changing to the left-hand lane, instead of to keep on default track. As *c1* is reaching the desired speed, the acceleration is decreasing. It happens in a linear fashion. The truck also reduces the acceleration linearly as the desired speed is reached. At time 8, the deceleration is slightly more accentuated due to the entrance of the *c1* on the default lane, as soon as overtaking is completed.

Analysis regarding to the time spent in overtake or performed all the route by each vehicle can also be computed. Figure 5.19 shows two examples of this verification type. These property using the reward “step”, responsible for providing the value 1 for each state change in the model, which is equivalent to 1 second in real scenery. The



(a) Movement in a normal overtake



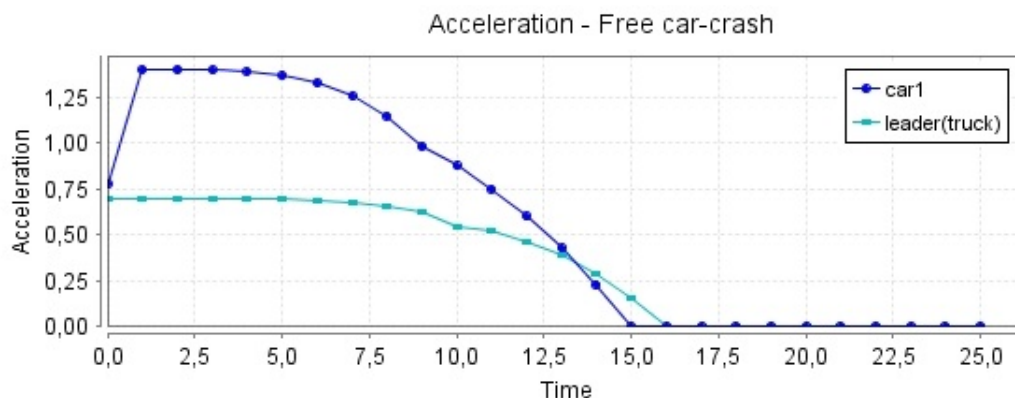
(b) Movement with accident

Figure 5.17: Scenario analysis

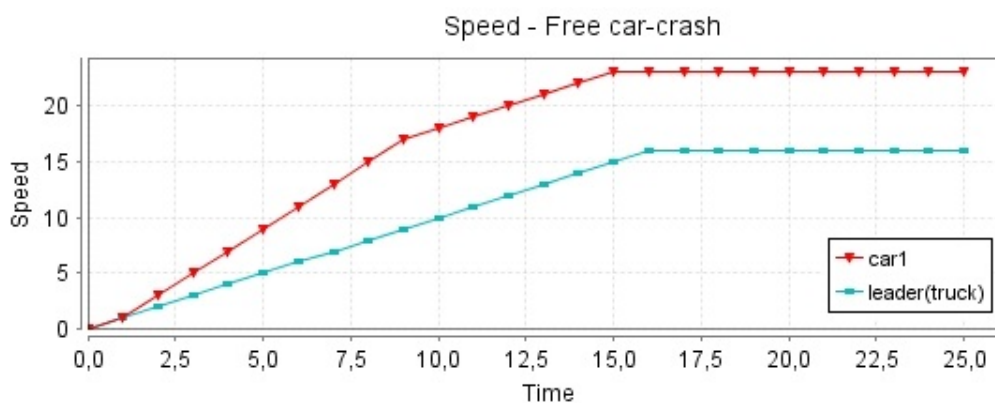
properties use the operator \mathbf{F} (Reachability “reward”), which associate the reward with each path of the model.

According to Kwiatkowska and Norman [2011], the reward property “ \mathbf{F} prop” corresponds to the accumulated reward along a path, until a satisfying property *prop* is reached. State rewards for the prop-satisfying state reached are not included in the accumulated value. In the case where the probability of reaching a state satisfying *prop* is less than 1, the reward is equal to *infinity*.

Property 8 calculates the overtake time, which results in collision for all possible initial states, thus the presented result was a value range of $[8.0, \textit{infinity}]$ seconds. The *infinity* value represents the initial states without collision. In other words, initial states that have probability less than 1. Therefore, to find the maximum time limit for the collision it is enough to analyze the log file which it will have the travel time for all initial states and their successors, that is available due to the parameter “print” in the *filter* command. However, the range of minimum and maximum time of collision



(a) Acceleration evolution



(b) Speed evolution

Figure 5.18: Acceleration and speed analysis in a free car-crash overtake

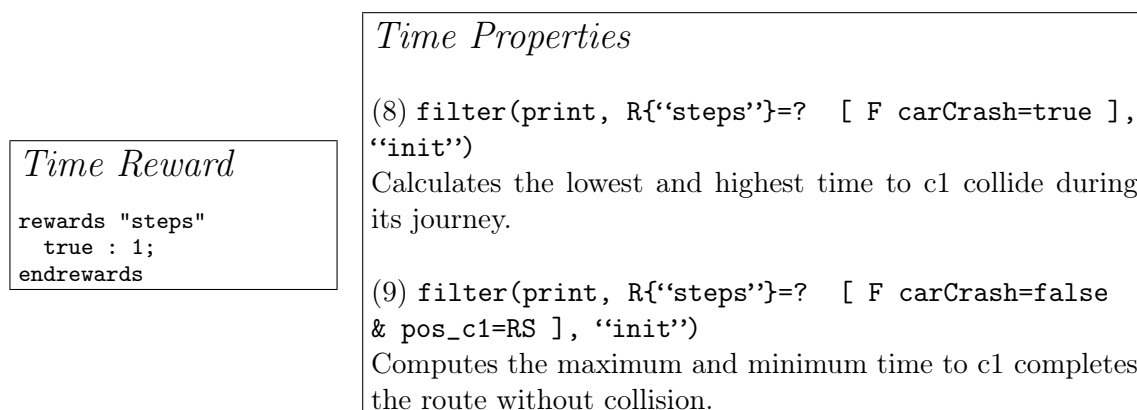


Figure 5.19: Time Rewards and Properties

are $[8.0, 11.0]$, in other words, the shortest time of accident is 8 seconds and the longest time happens at instant 11. Thus, they can be simulated, respectively with the initial states presented in Figures 5.20a and 5.20b. Furthermore, graphs as shown in Figures 5.17 and 5.18 can be generated for more detailed analysis based on these counterexamples.

Step		Mod_vC1	Mod_vC2	Mod_vL	Mod_dC1			Mod_dC2	Mod_dL
Action	#	v_c1	v_c2	v_l	pos_c1	lane	carCrash	pos_c2	pos_l
	0	1	6	1	2	1	false	6	20

(a) Minimum time of collision

Step		Mod_vC1	Mod_vC2	Mod_vL	Mod_dC1			Mod_dC2	Mod_dL
Action	#	v_c1	v_c2	v_l	pos_c1	lane	carCrash	pos_c2	pos_l
	0	0	0	1	1	1	false	1	19

(b) Maximum time of collision

Figure 5.20: Initial state for counterexample (unsuccessful overtaking)

In a similar way, property 9 calculates the minimum and maximum time for successful overtaking. This also takes into account all possible initial states. Thus, the range of values presented were among $[13.0, Infinity]$ seconds. Again, for all initial states in which the probability of \mathbf{F} to be satisfied is less than 1, it is assigned the *infinity* value. Thus, analyzing the PRISM log file in order to remove the infinity value, we can identify the following range of values $[13.0, 16.0]$, Figures 5.21a and 5.21b show their respectively initial states as counterexamples.

Step		Mod_vC1	Mod_vC2	Mod_vL	Mod_dC1			Mod_dC2	Mod_dL
Action	#	v_c1	v_c2	v_l	pos_c1	lane	carCrash	pos_c2	pos_l
	0	4	0	0	1	1	false	1	40

(a) Minimum time without collision

Step		Mod_vC1	Mod_vC2	Mod_vL	Mod_dC1			Mod_dC2	Mod_dL
Action	#	v_c1	v_c2	v_l	pos_c1	lane	carCrash	pos_c2	pos_l
	0	0	0	0	1	1	false	1	19

(b) Maximum time without collision

Figure 5.21: Initial state for counterexample (successful overtaking)

5.5 Conclusions

It is essential to test and analyze VANETs in order to prevent loss of life. Simulation is widely used to check new protocols and applications. However, the simulator has to

deal with two hitherto unconnected worlds, which must now work together (network and traffic). In this context, there are challenges that must be addressed by the research community. Nevertheless, an efficient alternative technique is the Model Checking.

In this chapter we presented the formal modeling and analysis of two models using Probabilistic Model Checking with different abstraction. The detail level can be different according to type of application. In order to verify performance in an Entertainment Application, a macroscopic vision could be enough. However, Traffic Safety Applications may require a more detailed analysis. Thus, some analytical formulas to represent position, speed, and even acceleration are necessary.

However, a macroscopic vision about a crossroad managed by a semaphore with different parameters was modeled. After, an overtaking vehicle scenario was implemented. The latter uses some analytical formulas to represent position, speed, and acceleration. This is done to show how our proposed modeling will be built. The first model suggests that prioritizing roads with heavy traffic can be a good alternative to improve traffic bottlenecks. This could increase traffic flow, improving the proposed Virtual Traffic Lights protocol. The other model shows a huge chance of an accident. But there are situations without collision.

In general, during implementation we noticed some limitations in language, for example, the absence of some mathematical functions and the lack of subroutine (function and procedure) and formal parameters. This fact impairs the legibility of the model and makes difficult the implementation/maintainability. However, the IDM and MOBIL model can be perfectly implemented and used in PRISM.

The implementation of motion provides important information such as instantaneous speed, acceleration and position through reward, besides answering questions regarding the probability that an event occurs. The modules responsible for the movement can be easily coupled with network protocols modules. In addition, the modeling is easily adaptable under various situations, such as multilane highway or an intersection. For example, to implement a curve road, simply, it is necessary to change the limited speed to a value less than a straight road, thus vehicles will reduce the speed. Thus, it abstracts itself for a certain distance that the vehicle is crossing a curve.

Nevertheless, through the coupling of movement modules presented in this study with models, which represent communication and signal propagation, it will be possible to do a full analysis of protocols and applications. In other words, VANET verifications will consider network, signal propagation and movement. Thus, protocols such as Virtual Traffic Light can be easily studied and through model checking advantages, as to be completely automatic and exhaustively, it will identify abnormal situations.

Chapter 6

Our guidelines to analyze VANETs in model checking

Outline. In this chapter we present our guidelines to model and analyze VANETs. Our work proposes the use of probabilistic model checking to perform a complete test. We are indicating some steps during the planning and a structure during the encoding, which includes mobility, communication and signal propagation aspects. Besides, an example of source code in PRISM language is presented.

6.1 Introduction

The modeling of a problem could be considered as an art. Researchers have to represent a system in full measure using abstractions, that is, they have to capture the main properties of a system using a language without extrapolating the limits that the technique requires it. Thus, it is necessary to remove irrelevant issues and abstract relevant details in a simplified form. These abstractions could be a problem, because sometimes it involves experience, which is gained through hands-on or studying adopted solutions and the available documentation. One problem is that model checking is still little used during analysis in VANET. Thus, researchers have difficulties in knowing which aspects need to be addressed and what the level of detail to treat those aspects is.

This work proposes guidelines to help researchers during analysis of protocol in VANETs. We want to assist beginners and experienced people. The guidelines are proposed taking as a basis our modeling experience from biological system to protocols in vehicular and computer networks [Braz et al., 2013b; Ferreira et al., 2015a]. Moreover, we also use some guidelines for simulation ([Barnett et al., 2012; Veysey and

Moran, 2013; EPEC, 2015]) and some technical standards documents [OMB, 2007]. Our purpose is to provide a modeling structure to help researchers focus on the most essential issues of the problem, that is, we are suggesting guidelines in order that the attention of analysis will be directed to the real problem. However, our structure is not meant to stifle the modeling, but instead, it raises questions that should be taken into account.

An introduction to the VANET was shown in Chapter 2, PMC was studied in Chapter 4 and a formal probabilistic analysis of a protocol in vehicular ad-hoc network is presented in Chapter 7. All these concepts are part of these guidelines, mainly the last one, where we have proposed a complete modeling structure which includes mobility, communication and signal propagation modules. As said before, it is necessary once communication could give new behavior or direction to the vehicles and the new ones could affect the network moving away or grouping the nodes, what interferes directly in the broadcast.

The **objective** of our modeling guidelines is to supply a consistent and plausible approach to the development of VANETs models in probabilistic model checking. The guidelines should be seen as a reference point and not as a rigid standard. They provide direction on the scope and common approaches to represent protocols/applications. The continual evolution of modeling through adaptation and innovation is encouraged.

The guideline provides an approach to model that is supported by a series of progression stages. Figure 6.1 illustrates the proposed modeling process. The feedback loops allow the process to go back to the prior stages. For instance, the researcher may judge that the model implementation is inadequate which can mean revisiting the design stage.

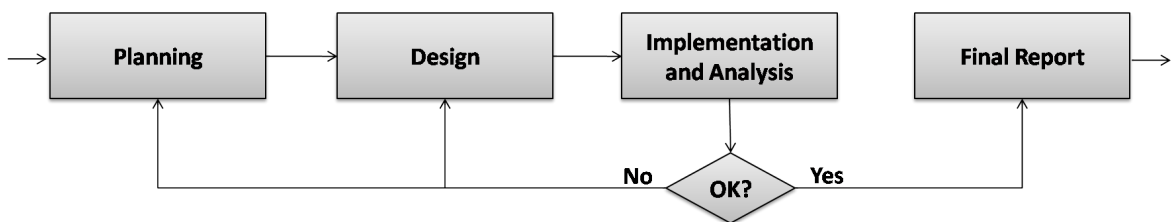


Figure 6.1: Guideline steps

In the **planning** stage, the researchers should agree about the model's intended use and the modeling objectives, in other words, what questions it should answer. At this point, it involves identifying and describing the processes that control or influence the vehicle's movement, the studied application and how the signal propagation is influenced.

The **design** and **implementation** stages involve a series of decisions on how to best implement the planning in a mathematical modeling environment. The decisions required at this stage include selection of the analytical models, selection of an appropriate granularity, besides the definition of the temporal representation used in the model.

At the end, a **final report** should describe the model and all information created through the modeling process. The report should be accompanied by all model files and supporting data. Thus, the results presented in the report can, if necessary, be reproduced as well as the model can be used in future studies.

Throughout the guidelines, key statements or paragraphs (of particular importance or interest) are presented in highlights or in boxes for added emphasis:

- a set of numbered guiding principles for the associated stage in the modeling process (starting with the first step-letter and numbered consecutively within each step) – see example below
 - **Guiding Principle P.1:** The modeling objectives...
- examples of concepts or principles (numbered consecutively within each step and provided in plain text boxes), for instance:
 - **Example 1:** Project objectives...

6.2 Planning

The process starts with planning stage, which focuses on gaining clarity about the problem and on the objective of the model. This involves all available data and knowledge of the product under study. It is never possible for one model to answer all questions on VANETs behavior. Thus, the model should be a goal.

Besides, the objectives should be used regularly throughout the analysis process as a guide to how the model should be designed, implemented and used to the tests. The main tasks in this phase are: **1)** to establish the context which the model development is being undertaken; **2)** to guide how the model will be designed, constructed and executed and **3)** to provide criteria for assessing whether the model is adequate for purpose and whether it has yielded the answers to the questions it was designed to address.

Moreover, at this stage, we have to keep in mind the weakness presented by the model checking technique, mainly the state-explosion-problem, which says that the

number of states in a model is exponential in the size of its description, it may just take too much time and typically also too much space to explore all these states. However, a wide environment, as a modeled city in model checking could be impossible, even using a computer with high processing power. Thus, specific scenario or a group of them, such as intersections, curves, streets or stretch of roads are more feasible. Hence, it is very important to determine the scenario's length and how many vehicles will pass through it. Therefore, we have formulated some guiding principles for planning a VANET model:

Guiding Principle P.1: Modeling objectives should be prepared early in analysis and they should state how the model could specifically contribute to the successful completion or progress of the protocol or application.

Guiding Principle P.2: Obtaining as much knowledge as possible about the studied product. In addition, getting earlier skills in analysis using Probabilistic Model Checking is very important.

Guiding Principle P.3: Target questions should be agreed upon and documented at an early stage of the project to help clarifying expectations.

Guiding Principle P.4: Which scenarios can reproduce the usage of product should be agreed and documented at an early stage.

Example 1: Project objectives and modeling objectives

Vehicular Warning System

Project objective: A collision avoidance system is an automobile safety system designed to reduce the severity of an accident. It uses radar and sometimes laser and camera to detect an imminent crash. Once the detection is done, these systems either provide a warning to the driver when there is an imminent collision or take action autonomously.

Modeling objective: Studying only the communication aspect, finding what is the best timeout to the packages to the safety channel.

Scenario: There are three vehicles involved. The car c_1 and a truck are in the same direction. The vehicle c_2 is coming in the opposite side and must be reported by c_1 about a wild animal presence in c_2 's lane on its next road curve. This scenario will happen in 200 meters.

Main question(s): Will the car be alerted in time?

6.3 Design

Design phase involves describing how the researcher intends to create and to represent a **conceptual model**, which is responsible for providing a description of suitable analytical models. One should think how they will be encoded and performed, as well as deciding on the model size, the measures standards, the type of probabilistic model and other aspects such as, the initial conditions, the parameterisation of the model and which model checker is the best to current analysis.

Conceptual model involves identifying and describing the processes that control or influence the movement and the signal propagation conditions. This conceptual document should consider the physical processes, in this regard, it provides information on how it is expected to impact on vehicles' acceleration and non-line-of-sight (NLOS) during communication. Moreover, our guidelines encourage regular reassessment of the conceptual model at all stages of the project. It is also encouraged to propose and maintain alternative conceptual model for as long as possible through the modeling project. In some cases, this may lead to the development and use of alternative mathematical models.

According to Barnett et al. [2012], a mathematical model describes the physical processes of a system using one or more governing equations. An analytical model makes simplified assumptions to enable solution of a given problem (e.g. a parameter called "free acceleration exponent" has a value to determine whether the gain of speed occur on a wet road).

Therefore, we have formulated some guiding principles for design a VANET model:

Guiding Principle D.1: The level of detail within the model should be chosen, based on the modeling objectives, the availability of processing power and its complexity.

Guiding Principle D.2: Analytical models could be considered to explore the significance of the uncertainty associated with different views of how the system operates.

Guiding Principle D.3: An architecture of modeling should be adopted. This architecture should represent the movement, the signal propagation and the network. Furthermore, it must have the goal of facilitating the changes of analytical models.

Guiding Principle D.4: The size, measures standards, the types of probabilistic models should be chosen to reflect the modeling objectives.

Guiding Principle D.5: Initial conditions and parameterisation should be thought.

Guiding Principle D.6: The model checker should provide sufficient features to be able to adequately represent all issues mentioned in the design's guiding principle.

Example 2: Design stage

Vehicular Warning System

Level of detail: Microscopic level – vehicles with realistic movements

Analytical models: The movement will be represented by Intelligent Driver Model [Kesting et al., 2010] and the propagation signal modeled by Nakagami model [Nakagami, 1960].

Standards: to mimic the protocol *802.11p* with rate of 100ms to broadcast beacon messages.

Architecture: Like proposed by Boban and Vinhoza [2011] (integrating movement, network and signal propagation)

Initial conditionals: Vehicles in opposite directions separated by 100 meters. All vehicles traveling at maximum speed of the road.

Model checker: Prism model checker (requirements: free license, probabilistic behavior, rewards and parametrized capability)

6.3.1 Type of models

Probability is an important component in the design and analysis of software and hardware systems. Traditionally, probability could be used to model unreliable or unpredictable behavior and with this purpose we have been proposing the usage of probabilistic model checking (PMC). This technique is able to calculate the likelihood of the occurrence of certain events during the execution of a system [Kwiatkowska et al., 2007].

PMC can be built using several types of probabilistic models. We have constructed our modeling through both important methods: discrete and continuous-time Markov chains (DTMCs and CTMCs). These methods were chosen because they implement computation tree logic (CTL) and their extensions with the reward operators. Thus, properties specifications are essentially identical to both methods.

DTMC admits probabilistic choices, in the sense that one can specify the **probability** of making a transition from one state to another. On the other hand, CTMC, frequently used in performance analysis, represents continuous real-time models and probabilistic choice: one can specify the **rate** of making a transition from one state to another. Probabilistic choice, in this model, arises through race conditions when two or more transitions in a state are enabled. Moreover, while each

transition between states in a DTMC corresponds to a discrete time-step, in a CTMC the transitions occur in real time [Kwiatkowska et al., 2007].

6.3.2 Level of detail

According to Levins [1966], during the models development, there is always a trade-off between realism, precision and generality, it is not possible to maximize all simultaneously. Thus, the design step involves simplifying the studied systems, which could be inherently complex, in order to mimic the key behaviors. However, there is no perfect way to simplify a problem, therefore, we are proposing a microscopic modeling to represent small scenarios, thus, we do not worry about path trip modeling (see Section 3.1). We are also abstracting origin, destination and the routes to reach their final address. Moreover, the standards to bring the vehicles into the models again within these small scenarios could be used to implement a traffic generator (Section 3.2.2).

Nevertheless, the modeling could adopt the concept map for realistic mobility models presented in Figure 3.4 to reach a good level of detail. The transitions between states representing realistic movement will be done through car following models. According to Section 3.2, this representation can model vehicle interaction, which includes the microscopic aspects, such as lane changing and decreasing/increasing the speed due to the surrounding traffic. For instance, the IDM model, which shows a crash-free collective dynamics, exhibits controllable stability properties and implements an intelligent braking strategy with smooth transitions between acceleration and deceleration.

External influences are the impact of VANET communication on the motion patterns. The signal propagation also is modeled using analytical models. We have been proposing the usage of Stochastic ones, since discussed in Section 3.2.2, they are more realistic. These models can distinguish cars of trucks, to apply different speed constraints and considering distinct topologies or clustering of vehicles, such as those introduced in Sections 5.3 and 7.1.

6.3.3 Initial conditional and limits

Another interesting issue is how the model will be initialized. The initial state is defined by the initial value for all variables and the researcher must be aware about this starting state. Typically, if the initial value of a variable is omitted, it is assumed to be the lowest value in the range (or false for a Boolean). However, it is necessary

to check this information or to make the initial value explicit by declaration. It is also necessary to define if there will be more than one initial state and whether the verifier supports this feature.

If on the one hand we must think about the initial conditions, on the other, the limits of the modeling should be discussed. We think that a clear definition of boundaries model are a key factor for achieving a reliable analysis. These limitations can be arising from insufficient information and mainly by the model incapacity to capture real-world complexity. In other words, there are limitations arising from simplifications imposed by the analytical models chosen. For instance, the classic implementation of IDM did not copy realistic response in case of critical situations such as collision, however, the extended versions solve this problem.

Furthermore, pedestrian and cyclist flows, for example, can limit the capacity for other road users through increased delays. The pedestrian volume and crossing times could impact the road user capacity and should ideally also be considered. The effects of pedestrians and cyclists must be included in the modeling, however, the IDM does not treat this question.

6.3.4 Measure units

As a last issue concerning the level of detail, we would like to highlight aspects about network throughput and measure units. An analysis could use different types of measure, for instance, applying kilometers to the position and hours to time or even centimeters and milliseconds for the respective greatness. It is reasonable to think the smaller the granularity, more accurate the model will be. However, we have to avoid the explosion of states during the verification in the meantime give a good accuracy. Thus, the researchers should to find a compromise, for instance, meters (m) and seconds (s) could be good units to be adopted by movement modules.

With regard to the network capacity, several studies to measuring wireless ones have been done [Teixeira et al., 2014; Lin et al., 2012; Neves et al., 2011], basically, they have tried to estimate the number of received package. The transmission capacity of these studies are relative to weather condition, the environment and the specified parameters according to the type of observed application. For example, we could consider a network with an average bitrate of $3Mbps$ in a VANET application, which exchanges packets data with 1,500 bytes. In this example we use a simple calculation to represent the amount of package at network:

$$\begin{aligned}
3 \text{ Mbps} &= 3,000,000 \text{ bits/s} \\
&= 375,000 \text{ bytes/s} \\
&= 250 \text{ packages/s}
\end{aligned}$$

Therefore, regarding a network composed by three vehicles, we could considerate that each node will transmit approximately 83 packages ($250/3$). If we are going to think in a transmission rate, a car transmits a package every $0.083 \text{ ms} - 83(\text{packages})/1000(\text{ms})$.

6.3.5 Architecture

Our guidelines propose a modeling with a microscopic focus. The idea is to describe the nodes' movement through analytical models. Furthermore, formulas represent signal propagation, which work according to the distance between the nodes. The communication will consider the signal propagation in a non-deterministic fashion. The movement could take into account the position, the speed and the acceleration.

Figure 6.2 shows an abstraction of the proposed idea. This model is divided into three wide groups (gear) implemented by modules and/or formulas in some model checker language. The groups exchange data among each other and it can change their behavior according to the interaction. We have been moving toward groups with higher cohesion and lower coupling to represent motion, signal-propagation and network. Thus, we can change parts of the modeling according to our need. For instance, change modules responsible for an urban street to a highway, or change the code responsible for signal propagation of expressions for the log-normal shadowing path-loss to Nakagami model ([Van Eenennaam, 2008]).

The black arrows (Figure 6.2) indicate the basic flow of information among the groups. The network (VANET application) will inform, for example, if the cars must move forward, change lanes or stop. This can be represented by Boolean variables. The movement modules will provide the distance between vehicles that will be accessed by the formulas of the propagation model, which will calculate the communication probability to the network group. Figure 6.3 depicts the same structure, however in a lower level of implementation (more details are presented in Section 6.4.1).

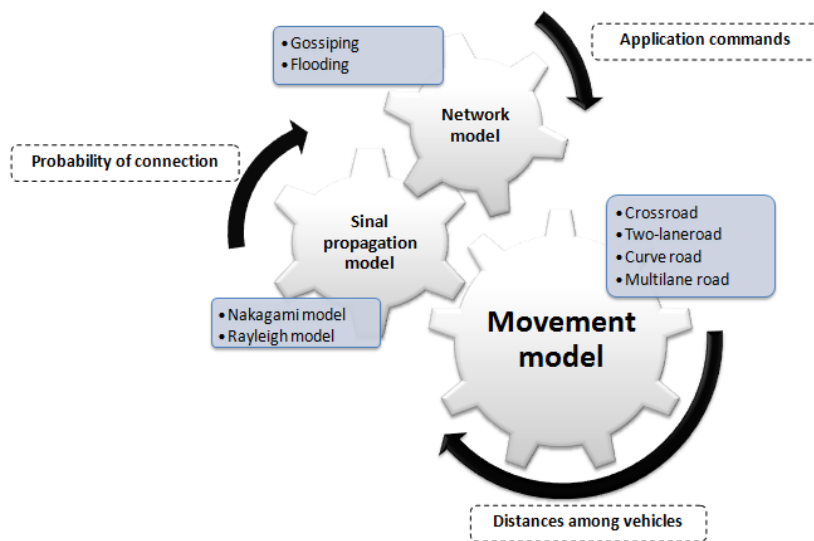


Figure 6.2: Proposed Architecture

6.3.6 Model checker

We believe that an important step in the modeling process is the model checker selection, in which all possible options are considered. There are a wide variety of available tools, with a number of different capabilities suited to different kinds of problems. However, according to Strunk et al. [2006] the variety poses two problems. First, it is difficult for researchers outside of these specific domains to understand what capabilities exist in practice. Second, while the variety is of great benefit to practitioners, it is intimidating to know which tool to choose for a particular problem, when no comprehensive discussion comparing and contrasting the different tools is available. Nevertheless, for our guidelines, the following features are useful when choosing a tool:

- Multiple user interfaces (graphical and prompt interfaces)
- Software license
- Portability
- Language properties

PRISM is free and it is a multi-platform. It has two versions of PRISM, one based on a graphical user interface (GUI) and the other based on a command line interface. Both use the same underlying model checker. The latter is useful for running large batches of jobs, leaving long-running model checking tasks in the background. However, GUI allow users to interact using several input tools, giving to the developers

more freedom to display the information, images and other elements. Furthermore, PRISM includes also a simulator, a tool which can be used to generate sample paths (executions) through a PRISM model. From the GUI, the simulator allows you to explore a model by interactively generating such paths. This is particularly useful for models' debugging during development and for running error checks on completed models [Kwiatkowska and Norman, 2011].

In addition to the desirable features of the tool, there are some critical resources to encode the presented architecture in Section 6.3.5, which are:

- Modeling probabilistic behavior
- Synchronization ability
- Parameterisation capability
- Multiple initial-states possibility
- Rewarded structures

In order to represent movement, signal propagation and network some features are required. For instance, the skill of supporting probabilistic behavior is necessary to represent the uncertainties of the propagation medium in wireless communication, thus, the messages delivery can be treated in a non-deterministic way. Furthermore, modules synchronization must also be present. Thus, we can force two or more modules to simultaneously update their variables, for example, the position and speed of vehicles should be changed together. Another important aspect is the capability of quantifying the behavior, such as rewarded structures offered by main model checkers.

Languages to model VANETs and to specify properties must have a good readability and writability for researchers with an intermediate level of knowledge in model checking. One of the most important criteria for judging a programming language is the ease with which programs can be read and understood. In this case, the readability is yet more significant, once, no specialists in model checking could use our guidelines. Thus, the language syntax must favor the understanding and the ease of maintenance. Moreover, we have to measure the ease of the properties specification language to argue the constructed model, i.e., the writability to specify the continuous stochastic logic, which are used to give information about the probability of an event's occurrence and to question about rewards/ cost of transitions or states.

According to Goguen [1984] the parameterized programming is a powerful technique for the reliable reuse of software. However, we ought as often as possible

parameterize the models and look for tools that allow such action. On the PRISM, the values of constants may be declared permanently in the source code or informed before execution via command line or GUI. This latter way to input values serves as parameters, which can be used to define the size of the variables or the initial values of them. Furthermore, constants can be used in conditional statements for state transitions, changing the model behavior. For example, a change of state depends on the value inserted in a constant called *protocol*, which could setup a flooding or a gossip behavior.

6.4 Implementation and Analysis

At this step, the design is encoded. Here, a model checker is required, in such way that verifications can be performed. In general, the step of implementation and analysis in our guidelines involves following the three basic steps of model checking: (1) specifying the properties that the system must satisfy; (2) constructing the formal model of the system, which should capture all the essential properties and (3) running the model checker to validate the specified properties.

Here we have to pay attention to the following principles:

Guiding Principle I.1: A model should be constructed according to the design, and documented as built.

Guiding Principle I.2: It is recommended and sometimes essential to construct the models by increments of versions.

Guiding Principle I.3: Adopting naming convention during encoding could improve the readability and consequently the maintainability.

Guiding Principle I.4: Following the basic architecture implementing the movement, data exchange and signal propagation designed.

Guiding Principle I.5: Performing basic tests to ensure that the model is working properly before the target questions.

Concerning the Principle I.1, like in the software development process, following the generated documents in all steps is very important. For example, to allow a substitute researcher in any eventuality or even to make transparent the testing and possible troubleshooting during the modeling. If it is necessary to do any adequacy in this implementation phase, then is important returning to previous stages and updating the documents.

Figure 6.1 shows the modeling as an incremental process, i.e., the planning, the design and implementation are revisited (changed) as more is learned about the system

(Guiding Principle I.2). Another factor responsible for the constant feedback loop is the size of the probabilistic model (the number of states/ transitions), which is critical to perform model checking, since both the time and memory required are often proportional to the model size. Unfortunately, it is very easy to create models that are extremely large. Below there are a few general tips for the reducing model size according to Kwiatkowska and Norman [2011], which we think it is interesting to reproduce in our guidelines.

- Look for variables that have unnecessarily large ranges and try to reduce them. Even if your model needs large variables, it is generally a good strategy to first get a smaller version building successfully and then scale it up afterwards.
- Similarly, can you (if only temporarily) reduce the number of modules/ components of your model? Start with the smallest number of components possible and then add others one by one.
- Do you have any inter-dependencies between variables? For example, perhaps you have some variables which are simply functions of other variables of the model. Even if these are convenient for model checking, they can be replaced with formulas or labels, which do not contribute to the state space.
- Do any variables include more detail than is necessary for the model? Perhaps this can be exploited in order to reduce the number of variables in your model.
- More generally, are any aspects of the model not relevant to the properties that you are interested in? If so, start with a simpler, more abstract version of the model and then add more details if possible.

About the verification of the implemented model, we initially advise the creation of simple properties responsible for answer basic questions, for example, to test whether the vehicles are respecting the speed limits and the proposed scenario and also if the analytical model that represents the signal propagation is producing the correct values. Furthermore, another technique aiming to improve the quality of the model, is to employ the simulation prior to perform the model checking. The simulation can be used effectively to get rid of the simpler category of modeling errors. Eliminating these simpler errors before any form of checking, may reduce the cost and time-consuming verification effort.

Regarding the Guiding Principle I.3, several researchers could read your model, thus, we ought to have in mind that a good code not only works, but it also must be

readable and easy to maintain. Proper naming is a critical component of making the code productive. For this reason, strict naming conventions are a core aspect of all coding standards. Adobe Resource Center [2016] gives advices about this issue and we think it is interesting to mention some in our guidelines:

- Limit your use of abbreviations – Use abbreviations consistently. An abbreviation must clearly stand for only one thing. For example, the abbreviation “Mod” might represent “module”.
- Concatenate words to create names.
- Use mixed-cases (upper and lower case) when you concatenate words to distinguish between each word for readability. For example, select “carCrash” rather than “carcrash”. Another solution it is to use underline to separate them.
- Name a module by describing the process or item, such as “scheduler” to a module responsible by organizing of different processes working in synchronism.
- Don’t use nondescriptive names for formulas or variables.
- Keep all names as short as possible.
- Remember to keep names descriptive.

In addition to naming conventions, try to repeat the same names to aspects that have already been modeled in another scenarios or studied protocols. For example, for all created scenarios use the constant RS (Road size) to parameterize the road length. Thus, the reader will be most comfortable when analyzing another code. In Example 3 is depicted a table of naming convention. This type of table could be attached to the final report in order to facilitate the study of the created model.

Example 3: Naming convention adopted		
Token	Pattern	Example
Formulas	employed the prefix: $f_$	$f_dist_c1_c2$
Constants	used capital letters	RS (Road Size), T (Time)
Vehicles	used the form: $\langle type \rangle \langle unique_id \rangle$ type=[c-“car”, t-“truck”, l-“leader”]	$c1, t1$
Modules	obey the form: $Mod_ \langle function \rangle _ \langle vehicle \rangle$ function=[spd-“speed”, pos-“position”, node-“network”]	Mod_spd_c1
Variables	employed the form: $\langle vehicle \rangle _ \langle descriptive_name \rangle$	$c2_crashed$

6.4.1 Encoding (Guiding Principle I.4)

Figure 6.3 shows an intermediate-level encoding of proposed architecture in Section 6.3.5. This figure shows the abstraction to one vehicle named *c1*. However, for other cars the coding is similar. Once most model checkers requires it, the first step is to choose which type of model will be used, stating for example, if the behavior is discrete or continuous. In this case, some properties could be enabled during analysis and the user should work with rate instead of probability. At our figure, **blocks** represents that a choice can easily be made without affecting drastically the code, for instance, to choice between *dtmc* or *ctmc*.

After, we are proposing a parameter section in the code. Since our solution uses analytical formulas, the modeling perhaps requests to evaluate the closed-form solution. In this case, all of the model parameters need to be set up appropriately. Thus, using a section, the user will always declare constants in the beginning of the file and use capital letters as naming conventions.

Three gears in the proposed architecture (Figure 6.2) are implemented by three different parts of the codes. Each part is bounded by dotted rectangles of different borders. The rectangles represent black-boxes. That is, the user can change some internal behavior, however, this will not affect the implementation of other parts. We can still change the whole rectangle (gear), for instance, if the researcher desires a different scenario with the same protocol and the signal propagation formula.

To maintain a high cohesion and a low coupling between the parts of the code, the coupling point has to be defined in each rectangle. These hook-codes are shown as the beginning of the arrows. For instance, the formulas “*f_prob_c1_c2*” and “*f_prob_c2_c3*” could be implemented by different blocks (analytical models), however the names should remain the same. Another example are the formulas that are used to give the distance among vehicles (lines 9-10). Thus, the user could represent motions with uniform (u.m.) or non-uniform movement (n.u.m), this latter, yet has the option to mimic the acceleration (acc.) in real way by analytical models such as IDM. However, one more time, the names of the hook-codes (*f_dist_1_2* and *f_dist_2_3*) should remain the same.

The arrows in Figure 6.3 also show the information flow among the parts of the code. The thinnest arrow shows that the probability of communication is calculated by considering the distance among the vehicles. In turn, the smallest arrow (below) shows that the protocol under study will consider whether a message arrived, according to the probability provided. Finally, the vehicle will move according to commands passed by the protocol (longest arrow).

```

1 //-----Parameters----- dtmc or ctmc
2 //-----Parameters-----
3 const int T = 1; //s
4 const int RS = 500; //m (ROAD_SIZE)
5 const int DESIRED_SPEED_CAR = 7; //m/s
6 //-----Movements-----
7
8 //Absolute distance between vehicles
9 formula f_dist_1_2 = abs(pos_c1 - pos_c2);
10 formula f_dist_2_3 = abs(pos_c2 - pos_c3);
11
12 formula f_m_c1 = Uniform movement or Acceleration with:
13 formula f_v_c1 = IDM model or Gipps model
14
15 module Mod_vC1
16     v_c1 : [0..DESIRED_SPEED_CAR] init 1;
17
18     [m] (pos_c1 <= RS) & (v_c1 <= DESIRED_SPEED_CAR) -> (v_c1' = f_v_c1);
19 endmodule
20
21 module Mod_dC1
22     pos_c1 : [1..RS] init InitPosC1;
23
24     [m] pos_c1 <= RS -> 1: (pos_c1' = min(RS, pos_c1 + f_m_c1));
25 endmodule
26
27 //-----Network-----
28 module Mod_nodeC1
29     active : bool init true;
30
31     [n]
32     graph TD
33         A((A)) --> B((B))
34         A((A)) --> C((C))
35         B((B)) --> C((C))
36         C((C)) --> D((D))
37         D((D)) --> C((C))
38     end
39 endmodule
40
41 //-----Signal Prog.-----
42 formula f_prob_c1_c2 = Nakagami or Path-loss
43 formula f_prob_c1_c3 =
44 //-----Scheduler-----
45 module Mod_Scheduler
46     turn : [0..T];
47
48     [n]
49     [m]
50 endmodule
51
52 //-----Rewards structure-----
53
54 rewards "probability_1_2"
55 true : f_prob_c1_c2;
56 endrewards

```

Figure 6.3: Encoding of Proposed Architecture

The first block in Figure 6.3 represents the movement using two modules synchronized per vehicle (label “ m ”), because the position and the velocity should be updated in the same time. The second rectangle represents the studied protocol. The number of modules by vehicles could vary according to the studied application. Here, the directed graph represents the different states which a protocol can admit, an usual form to represent it, as shown by Figures 3.9, 3.10b. It is important to remember that the amount of modules in this part is also relative to the number of vehicles. The last box is usually depicted by formulas regarding to the signal propagation, which will depend on the selected analytical model.

The last module, named Scheduler, is essential to our architecture, once is responsible for alternating the state change responsible for representing the movement and the sending of the messages. For instance, during “ T ” time unit, a vehicle can move once and each vehicle is able to send 3 packages. In this module, the discrete and continuous implementation model are different, by the fact that the first one works with probability and the second with rates. See an example of the code in Example 4.

Example 4: Rates module – synchronizing CTMC modules in Prism Language

```
const double c1_c2_send_rate = T/0.333; // (sent time is 0.333 seconds or 3 msgs/s)
const double max_send_rate = T/13.50; // (mean inter-arrival/sent time is 13.5 seconds)
const double movement_rate = 1/T; // (movement rate by T second)

module Mod_Scheduler
  [n1n2] true -> (c1_c2_send_rate*prob_c1_c2 <= 0 ? max_send_rate :
                 c1_c2_send_rate*prob_c1_c2) : true;
  [n2n1] true -> (c1_c2_send_rate*prob_c1_c2 <= 0 ? max_send_rate :
                 c1_c2_send_rate*prob_c1_c2) : true;
  [m] true -> movement_rate : true;
endmodule
```

Example 4 shows the dynamics of the scenarios, which has two vehicles moving and sending messages. Its motion rate is defined by the constant *movement_rate* and the broadcast between *node₁* and *node₂* obeys the *c1_c2_send_rate*. A common technique, as used here, is to make one action passive, with rate 1 (other modules) and one action active, which actually defines the rate for the synchronized transition (carried out by *Mod_Scheduler*).

Figure 6.4 depicts a random path generated by Prism during the execution of Example 4. This path exemplifies our architecture. The first column shows which block of code is responsible for the state transitions in the current step, as soon as the motion is performed. Then, the network exchanges information according to the rate, which is influenced by the probability of communication. The third column shows

the transition time, which is randomly generated in this path during the information exchange. Moreover, according to the rate, the movements occur about every one second. The other columns are exclusive on this example, they report when the vehicles have a message to be computed.

Step		Time	Mod_nodeC1	Mod_nodeC2	Rewards	
Action	#	Time (+)	c1_receivedC2	c2_receivedC1	["C2receivedFromC1"]	["C1receivedFromC2"]
	0	0,	0	0	0,	0,
[m]	1	0,010356	0	0	1,	0,
[n1n2]	2	0,128792	0	1	1,	0,
[n1n2]	3	0,839907	0	1	0,	1,
[n2n1]	4	0,851022	1	1	0,	1,
[n2n1]	5	0,862137	1	1	0,	0,
[m]	6	0,973287	1	1	0,	1,
[n2n1]	7	1,18444	1	1	1,	0,
[n1n2]	8	1,29559	1	1	1,	0,
[n1n2]	9	1,40674	1	1	0,	1,
[n2n1]	10	1,65791	1	1	0,	1,
[n2n1]	11	1,80908	1	1	1,	0,
[n1n2]	12	1,96025	1	1	0,	0,
[m]	13	2,07142	1	1	0,	1,

Figure 6.4: Execution path of the proposed architecture

6.4.2 Basic analysis (Guiding Principle I.5)

The model ends with the rewards definitions, which allows the quantification of different aspects of the model, such as states and transitions. The possibilities of analysis using rewards and other operators are countless in a modeling with the proposed architecture. They can be made concerning to the movement, the network or signal propagation. Furthermore, they may involve more than one gear of our structure, for instance, finding out how many messages are received by the vehicles in a particular distance.

Figures 6.5–6.10 show basic rewards which can be included in the model, such rewards are used, for instance, to check whether the movement and the acceleration are proper, in addition to analyzing the probability and the rate of the messages sent in order to ensure they are consistent with the distance between vehicles. Finally, the messages sent by network modules could be computed as well.

The verification will depend on the analysis purpose and the studied scenario. However, in this sub-section, we have proposed some simple queries which can be performed to check whether the model is working correctly. These issues are:

1. Verify the vehicles' acceleration.


```

rewards "acc_c1"
  true : f_acc_c1 >= 0 ? f_acc_c1 :
        (-1*f_acc_c1)+1000;
endrewards

```

Figure 6.5: Acceleration reward

```

rewards "rate_c1c2"
  true : (c1_c2_send_rate*f_prob_c1_c2 <= 0 ?
        max_send_rate :
        c1_c2_send_rate*f_prob_c1_c2);
endrewards

```

Figure 6.6: Rate reward

```

rewards "probability_1_2"
  true : f_prob_c1_c2;
endrewards

```

Figure 6.7: Probability reward

```

rewards "spd_C1"
  true v_c1;
endrewards

```

Figure 6.8: Speed reward

```

rewards "mov_c1"
  [m] true : f_mu_c1;
endrewards

```

Figure 6.9: Position reward

```

rewards "C1receivedFromC2"
  [n1n2] true : 1;
endrewards

```

Figure 6.10: Received msgs.

2. Verify if the vehicles are obeying the speed limit and the acceleration.
3. Verify if the new positions are related to the speeds.
4. Verify if the rates and probabilities are according to the distances among vehicles.
5. Verify if the messages are sent/received according to the rates and probabilities.

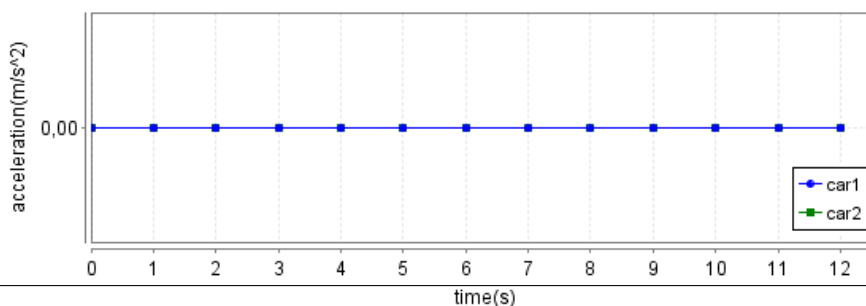
We can apply these rewards at the same scenario presented in Example 4, where two cars moving at constant speed in the same direction. The vehicle *c1* is 1 *m/s* faster than *c2* and this latter one is initially six meters ahead. The communication range is only 5 meters and they broadcast message in one-hop protocol. However, Example 5 shows the minimum queries to analyze the correctness of a model using our architecture and their respective results.

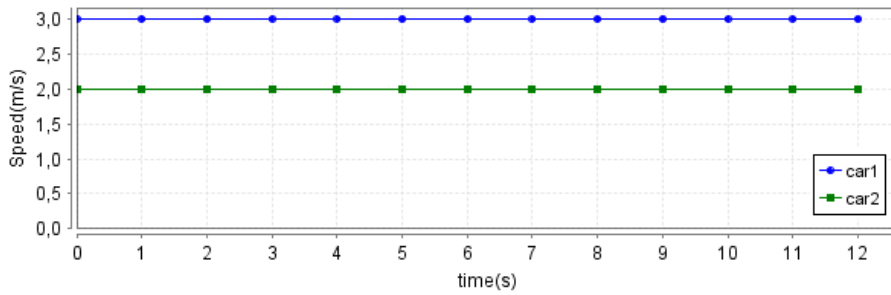
Example 5: Testing the basic structure in Prism language

–Verifying the vehicles' acceleration and speed:

Property I – $R\{\text{"acc_C1"}\}=?[I=\text{time}]$

Property II – $R\{\text{"spd_C1"}\}=?[I=\text{time}]$

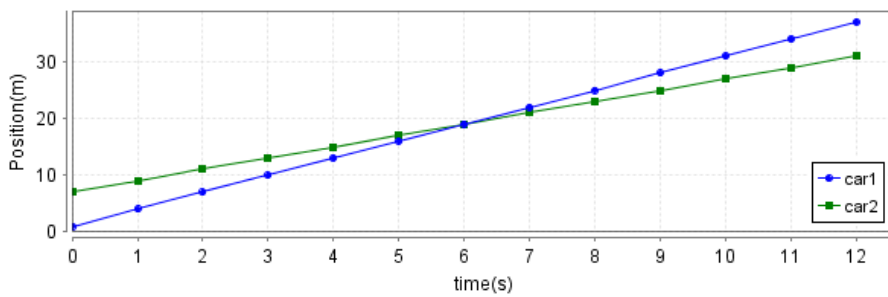




Graphs above show the expected behavior to uniform movement, where the acceleration is zero to the vehicle *c1* and *c2*, i.e., the vehicles maintain their constant speeds, such as proposed by Example 3.

-Verifying the vehicles' movement:

Property I – $\text{InitPosC1} + R\{\text{"mov_c1"}\} = ? [C \leq \text{time}]$

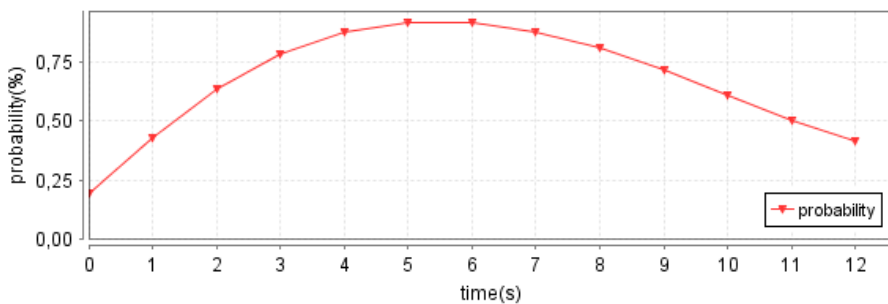


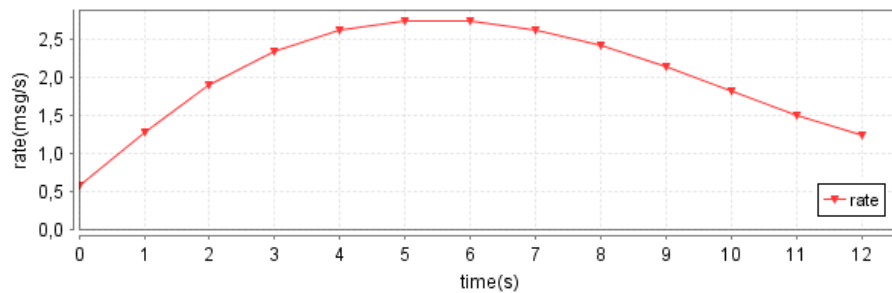
Graph above shows the vehicles' position, where the movement is compliant to the speed. Vehicle *c1* is 1 *m/s* faster than *c2*, which is overcome in the instant 6.

-Verifying the signal propagation:

Property I – $R\{\text{"probability_1_2"}\} = ? [I = \text{time}]$

Property II – $R\{\text{"rate_c1c2"}\} = ? [I = \text{time}]$

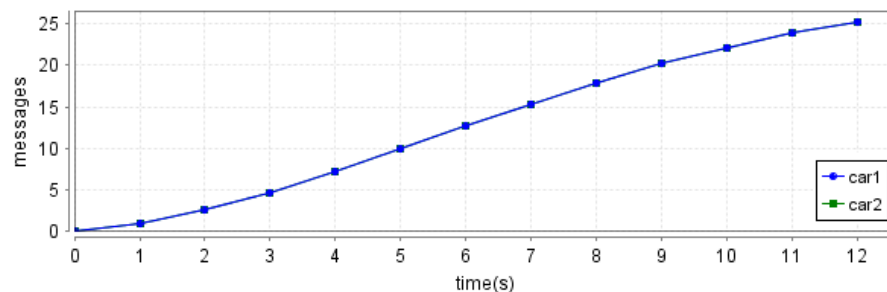




First chart above shows the vehicles' connection probability, which is a direct consequence of the distance between $c1$ and $c2$. Note that the connection possible is certain when the vehicles are closer, reaching 100% when both vehicles have the same position at the instant 6. Rate of broadcast between $c1$ and $c2$ is depicted on second graph, which has the same curve of the first one. When the probability is 100% the network throughput is all used (3 msgs by second), on the other hand, when the vehicles are further the throughput is closer to zero.

–Verifying the network:

Property I – $R\{C1receivedFromC2\}=? [C \leq time]$



Last graph shows the cumulative number of messages received by $c1$ and $c2$. Near to the instant 6 the number of messages has a fastest growing. At the beginning and end of the line, are required three seconds to broadcast about 5 packets. Therefore, obeying the rate of throughput.

6.5 Final report

The final reporting connects all communication during planning, design and construction. The model, all the collected data and the information generated through the analysis process need to be archived in the report, so the results presented can be reproduced and the model can be used in future studies. Besides, it may be interesting

to do a staged reporting, which implies writing progress reports with researchers after each stage in the project.

As said, the modeling process collects and generates a large amount of data and information. The visualization of such dataset can be a hard task, but it is essential to the communicating to report the model results. However, the researchers are encouraged to use traditional graphics, such as time series, scatter plots and maps/diagrams of conceptual models to let the communication more suited and intuitive. Nevertheless, our guidelines to this step are:

Guiding Principle R.1: The reports should be prepared following all the stages.

Guiding Principle R.2: Model data and results should be presented using clear approaches to visualization.

Guiding Principle R.3: A model archive should be created to allow the model results to be reproduced exactly.

The archiving has a dual goal; first, it must allow for exact reproduction of the results presented in the model report, and, second, it serves as a repository for all data, information and knowledge accumulated through the modeling process to facilitate future analysis of the VANET applications.

The steps of these guidelines can be used as a template for reporting the analysis. Table 6.1 gives an example of a model-report structure. It is important to remember that, as our guidelines, these are suggestions and it is up to the researchers to decide the necessary sections.

Table 6.1: Proposal to final report structure

Item	Title	Description
01	Title report	The title should reflect the project following by author's name and the date of production.
02	Contents	A table of contents including the titles or descriptions of the first-level headers.
03	Planning	All issues addressed in this step, e.g., objectives and scenario.
04	Design	All issues addressed in this step, e.g., types of probabilistic models and which model checker.
05	Implementation	All issues addressed in this step, e.g., which analytical models will be used and the naming convention adopted.
06	Conclusions	Deduction of model findings and recommendations for further analysis.
07	References	Full references of cited literature.
08	Appendices	Models and other files created during modeling which are important to the understanding of the analysis.

6.6 Conclusion

We have presented our modeling guidelines to supply a consistent and plausible approach to the development of VANETs models in probabilistic model checking. The guidelines are a point of reference and not a rigid standard. In other words, it provides direction on the scope and common approaches to represent protocols/applications. Moreover, a continual evolution of this guidelines is encouraged.

Our guidelines provide a sequence of steps where important issues which the researchers should not omit have been defined. During the design phase we have proposed an architecture that includes microscopic aspects to represent movement, network and signal propagation. Thus, our modeling structure helps the researchers to focus on the most essential issues of the problem.

The created model following our guidelines can explore information regarding to movement and communication. For instance, find the distance out until a message is received or a probability of connection among vehicles. By following our steps relevant issues could be raised regarding to which model checker is better, the importance of measures standards and thinking, as soon as possible, questions like: “What are the objectives?” and “What scenario could be used during the tests?”. Moreover, the guidelines show a flow of actions and what the minimum necessary to document all the analysis is.

Our architecture cares about low coupling of modules that represent movement, network and signal propagation. Thus, we can reimplement some of them and reuse others. This makes future analysis easier, because there will be a know-how about what aspects we must model and a knowledge base could be created. Furthermore, the studies obeying a same level of specification can be compared in an easier way. Thus, our guidelines are aligned with our goal to improve the quality of the analysis in VANETs.

Chapter 7

Our Complete Model to Verify VANETs

Outline. In this chapter we exemplify our proposal for the formal verification of VANET in a complete way. This is done through one model using analytical formulas in PRISM language. We have used the modeling structure proposed in our guidelines which includes mobility, communication and signal propagation modules. We present an analysis of a Vehicular Warning System involving three automobiles. The case study shows the influence of the initial positions, speed and timeout on communication.

7.1 Proposed Method

We have proposed a structured model which will guide future analysis in VANET. This model is divided in three wide groups composed by modules and/or formulas. However, there are modules responsible for node's movement and others ones by messages transmissions. The groups exchange data with each other and can change their behavior, for instance, a message requiring that the vehicle stop.

Our model was created with a microscopic focus. The idea is to describe the nodes' movement through the Equations 3.1 and 3.2. The signal propagation has been represented using analytical formulas (Equations 3.16 and 3.17), which work according to the distance among the nodes. Thus, the communication will be considered the signal propagation in a non-deterministic fashion. However, the interaction between traffic and communication systems can be represented more realistically. Furthermore, our microscopic model takes into account position, speed, acceleration and the data exchanged among the vehicles.

Figure 7.1 illustrates a proposed scenario to demonstrate the feasibility of the method. There are three vehicles involved. The car $c1$ will overtake the truck, called **Leader**, which travels slower. The vehicle $c2$ is coming in the opposite direction and must be reported by $c1$ about a wild animal presence in $c2$'s lane. This scenario will happen in 200 meters. However, the model must answer questions such as “Will the car be alerted in time?”.

An interesting feature of the model is that it does not have a specific initial state. The restriction implemented specifies only that vehicles $c1$ and $c2$ are in opposite directions, they are separated by RS meters and there is a *leader* between them. However, the leader position and the initial speed of all involved can be a combination of values. This creates several scenarios to be explored. The initial state also defines that there are no vehicles transmitting a message.

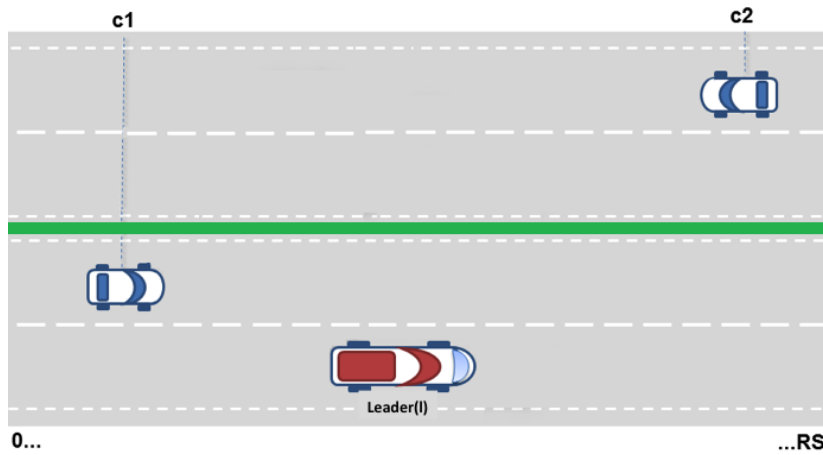


Figure 7.1: $c1$ overtakes the **Leader**, while $c2$ comes in the opposite direction.

The model fragment in Figure 7.2 shows the modules representing the position and speed of vehicle $c1$. The first command of module Mod_dC1 describe the changes of $c1$'s position given by $x = x_i + vt + (a/2)t^2$. Each transition of the model represents a time period that is defined by the constant t . The acceleration was calculated by the IDM (Section 3.2) which was implemented by formula a_c1_free . Since there are no obstacles ahead just the free-road acceleration strategy was considered. Similarly, module Mod_vC1 gives new speed by $v = v_i + at$.

The Network model was inspired by Boulis et al. [2008]. However, the nodes in VANET are always active. Thus, they receive or send messages all the time. We are considering a 1-hop flooding protocols (gossiping-style can be easily adapted). In this scenario, beacons messages are not modeled and the analysis considers only emergency ones. The vehicle $c1$ is the source node and it sends messages according to a *timeout*. The other nodes (leader and $c2$) only receive and forward.


```

formula a_c1_free = AM_car - AM_car * pow(v_c1 / desired_speed_car, exponent);
formula muv_c1 = (v_c1 + ( a_c1*pow(time,2)) / 2) > 0 ?
                (v_c1 + (a_c1*pow(time,2)) / 2) :
                (-1 * (v_c1 + (a_c1*pow(time,2)) / 2));

module Mod_vC1
  v_c1 : [0..desired_speed_car]; // speed in m/s

  [m] (pos_c1 <= RS) & (v_c1 <= desired_speed_car) ->
      (v_c1' = min(max(ceil(v_c1 + a_c1)*time,0), desired_speed_car));
endmodule

module Mod_dC1
  pos_c1 : [1..RS]; //(Road Size)    position in meters

  [m] (pos_c1 <= RS) -> (pos_c1' = min( ceil(pos_c1 + muv_c1),RS) );
endmodule

```

Figure 7.2: Movement modules implementation in the PRISM language.

Figure 7.3 depicts a fragment of Network group. Each module has a variable `send_i` which indicates whether the vehicles have messages to be transmitted. If the value is one, then the node received a message and has to broadcast it. Otherwise, the vehicle did not receive messages or already sent it. The variable `time_c1` is responsible to indicate the period for the next transmission. The constant `Tout` indicates the maximum *timeout* for this forwarding.

```

module node_c1
  time_c1: [1..Tout];
  send_c1: [0..1];
  [n] (send_c1=0)&(time_c1=Tout) -> (send_c1'=1);
  [n] (send_c1=0&pos_c1>=RS)|(send_c1=0&time_c1<Tout) -> (send_c1'=0);
  [n] (send_c1=1) -> (send_c1'=0);
  [s] time_c1=Tout -> (time_c1'=1);
  [s] time_c1<Tout -> (time_c1'=time_c1+1);
endmodule

module node_l
  send_l: [0..1];
  [n] (send_l=0)&(pos_l<RS) -> recvp1: (send_l'=1)+(1-recvp1):(send_l'=0);
  [n] (send_l=0)&(pos_l>=RS) ->(send_l'=0);
  [n] (send_l=1) -> (send_l'=0);
endmodule

```

Figure 7.3: Network module implementation in the PRISM language.

Signal propagation is implemented as the probability p of a PRISM command. In Figure 7.3 the formula `recvp1` formalizes this behavior. We are following the implementation proposed by Boulis et al. [2008] where it is necessary to compute the transmission powers, the signal-to-noise ratios and thresholds for each node. The difference is that we have computed **dynamically** the $rx_{i,j}$ from Equation 3.12, according to the movement of the vehicles instead of constant declarations.

Interactions between multiple modules, i.e. simultaneous changes in their state,

are modeled using synchronization, which is specified by *sync* labels (placed inside the square brackets) in front of guarded commands. In our model the Movement modules are labeled with “m”. The Network modules use the “n” label and additional modules take “s”.

Finally, Figure 7.4 shows the module responsible for alternating the group execution. First the movement is executed, then the network and finally some operations such as incrementing the timeout. The last one is necessary so that the link-probability is not affected by other state transitions.

```

module scheduler
  turn: [1..3]; // init 1;
  [m] turn=1 -> (turn' = 2); //movement
  [n] turn=2 -> (turn' = 3); //network
  [s] turn=3 -> (turn' = 1); //setup
endmodule

```

Figure 7.4: Synchronize Module implementation in the PRISM language.

In order to quantify the model, we have added rewards, which count each time when a condition is true. Figure 7.7 and 7.8 show rewards able to quantify the number of messages sent by *c1* and the receiving probability of *c2*, respectively. Figure 7.9 shows the code, responsible for providing a distance traveled by *c2* for each change of state in the modules labeled by “[m]”. The formula *muv_c2* is similar to the presented in Figure 7.2. Reward for *c2*'s acceleration and the power transmission between nodes *c1* and *c2* are represented by Figures 7.5 and 7.6 respectively. PRISM language does not support negative rewards, thus, we have added an overhead value in negative amounts which the properties must consider.

```

rewards "aCar2"
  true : accC2>=0 ? accC2 :
          (-1*accC2)+1000;
endrewards

```

Figure 7.5: Acceleration reward

```

rewards "Pc1_c2"
  true : rxDB_c1_c2<0 ?
          -1*(rxDB_c1_c2)+1000 :
          rxDB_c1_c2;
endrewards

```

Figure 7.6: Power Transm. reward

```

rewards "sByC1"
  [m] send_c1=1:1;
endrewards

```

Figure 7.7: Sent msg reward

```

rewards "recvpC2"
  true : recvpC2;
endrewards

```

Figure 7.8: Prob. reward

```

rewards "muvCar2"
  [m] true : ceil(muvc2);
endrewards

```

Figure 7.9: Distance reward

7.2 Results

In order to analyze some situations about the proposed scenario, several interesting questions can be made and some parameters like the *Timeout* (T_{out}) can be explored. We have divided the analysis by group to facilitate its understanding.

7.2.0.1 Propagation model

The model can supply the power transmission behavior or the communication probability between $c1$ and $c2$. Two properties below show their formal specification, respectively¹:

$$I) \nabla R\{\text{'Pc1_c2'}\}=? [I=t]$$

$$II) R\{\text{'recvpC2'}\}=? [I=t]$$

Figure 7.10 shows the minimum and maximum behavior over time for the first property. These two extremes are given by different initial states and are related to the approach speed between $c1$ and $c2$. As the vehicles are further away, the transmission power is weaker, when they are closer force is stronger.

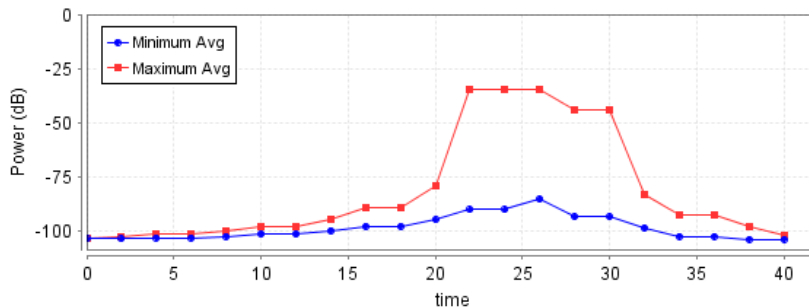
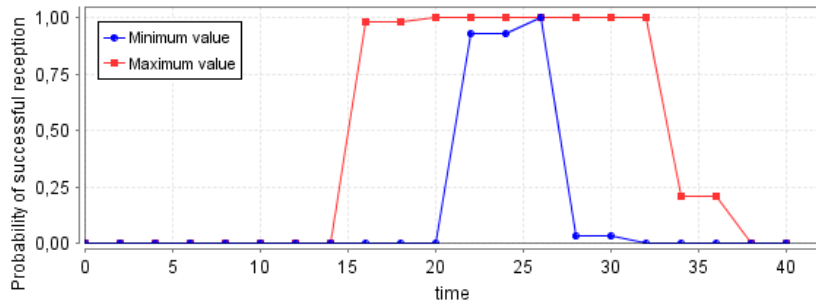


Figure 7.10: The average minimum and maximum of power transmission.

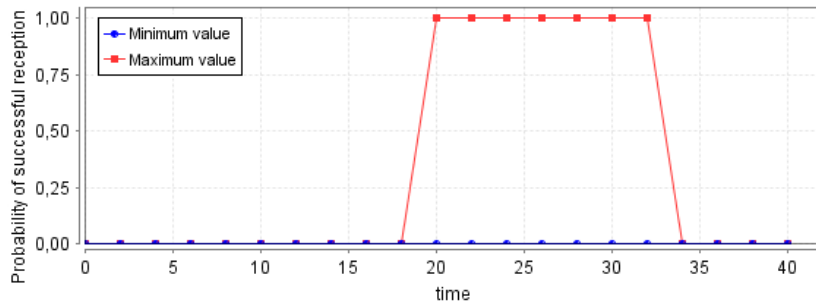
Property II shows three different ways of transmissions from $c1$ to $c2$. The first one, only $c1$ broadcasts the message and the sending is performed all the time ($T_{out}=1$). The next one has the same T_{out} , however *leader* sends other types of messages (i.e., there is noise). The last situation obeys the $T_{out}=2$ and the Flooding protocol (nodes broadcast the same messages). Figure 7.11 shows the minimum and maximum probability of the three behaviors in the order presented here.

Some comparisons can be done: at the beginning of the vehicles approaching, $c2$ does not receive messages because of the distance between nodes. The remoteness

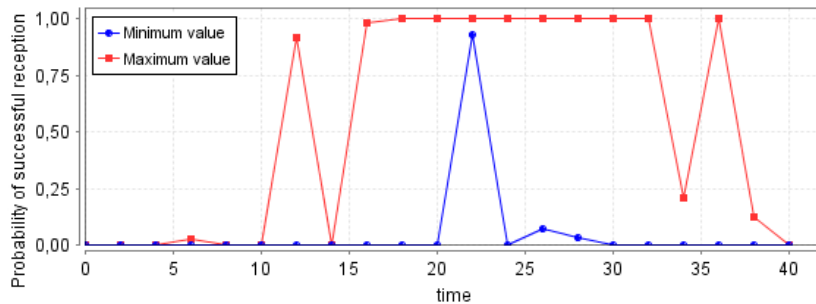
¹Symbol ∇ represents the condition evaluation:
 $(property > overhead)?(-1 * property - 1000) : property$



(a) Probability of successful reception when only c1 sending.



(b) Probability of successful reception when c1 and c2 sending.



(c) Probability of successful reception in flooding protocol.

Figure 7.11: Signal propagation analysis of c1 and c2 communicating.

causes the same behavior. The Graph 7.11a shows a higher probability to receive the message than 7.11b, because the noise disturbs the signal. The Graph 7.11c has the best maximum reception probability because the *leader* broadcasts the packet received by *c1*. Thus, the probability of *c2* receiving the message is higher. The oscillation in this case is happening due the *timeout* and the synchronous behavior of the modeled nodes. However, the Graph 7.11a presents the best minimum probability, due to the absence of noise and *timeout* oscillation.

7.2.0.2 Network model

Regarding messages, we have decided to verify what is the ratio between *c1*'s sent messages and *c2*'s received ones (*Property III*). Another question is (*Property IV*):

what is the probability that $c2$ receives a message directly from $c1$? This last query shows only a curiosity, because the more important is $c2$ to be alerted, regardless of who sent the message. Thus, the main question is (*Property V*): what is the probability that $c2$ receives a message? Three properties below show respectively these formal specification ²:

III) $R\{\text{"rByC2"}\}=? [F \text{ ‘‘finished’’}]/R\{\text{"sByC1"}\}=? [F \text{ ‘‘finished’’}]$

IV) $P=? [F (\text{send_c1}=1\&\text{send_l}=0\&\text{send_c2}=0) \& (X (\text{send_c1}=0\&\text{send_c1}=0\&\text{send_c2}=1))]$

V) $P=? [F (\text{send_c2}=1)]$

Table 7.1 shows the results for *Property III*. We can notice that *Tout* set up to 4 is a fine option since the received maximum ratio is the highest, due to the reduced broadcast and it has a good lower bound when compared to other results.

Table 7.1: Property Results

Timeout	Property III	Property V	Property VI
2	[0.28, 0.56]	[1.00, 1.00]	[25, 103]
3	[0.20, 0.75]	[0.99, 1.00]	[35, 103]
4	[0.25, 0.99]	[0.87, 1.00]	[48, 137]
5	[0.01, 0.79]	[0.02, 1.00]	[48, 84]

Property IV resulted in the range of $[1.03E-5, 1.0]$, which is the probability over all initial states. This property becomes interesting when we study the counterexamples created. After their analysis, we notice that the leader’s initial positions are related with the results, as well as the speed between $c1$ and the *leader*. The further away the leader begins, greater the probability of directly communication. It seems obvious because the leader will cross $c2$, thus, $c1$ and $c2$ can transmit directly. In other case, the $c1$ ’s speed is high enough to overtake the leader and they will communicate as desired. However, we can find through PRISM the best and worst situation to happen this data exchange.

Figure 7.12 show some analysis over the results to *Timeout=2*. It represents a probability of direct communication according to leader initial positions which varies from 17 to 200 meters, once that the scenario assumes a fixed position to the others involved. Thus, it was necessary to find the best initial speeds to the vehicles. The

²“finished” represents $(\text{pos_l}=\text{RS}) \& (\text{pos_c1}=\text{RS}) \& (\text{pos_c2}=\text{RS})$

graph shows the best probability when $c1$, $c2$ and $leader$ assume the respective initial speeds 5, 0 and 0 m/s . The worst case is given by values 3, 5 and 5 m/s , in same order. This last does not contain the minimum probability and we traced the two situations which include this value which have the speeds 3, 1, 0 and 4, 0, 1 in m/s . The series are coincident and the minor value of probability is obtained when the $leader$ starts in position 62 in both cases.

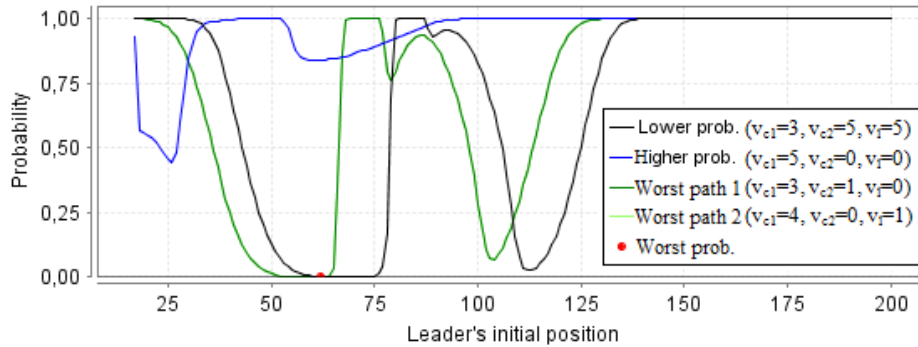


Figure 7.12: Probability of direct message reception

Table 7.1 also shows the results of *Property V*. The lowest *timeout* is sure about the warning. Setting the period of message to 3 we have the same probability with a lower traffic of messages. However, there is a higher chance of $c2$ not receiving the warning if the last two values are used. Analyzing the result we noticed that the average is lower to the $timeout = 4$ when the leader is in an initial position above 100 meters. For example, the values 3, 5, 3, 1, 1, 197 to the respective variables v_c1 , v_c2 , v_l , pos_c1 , pos_c2 , pos_l when the probability is about 0.93. The results to the higher *timeout* are also influenced by the distance of the leader, but also by the large period of time to send the next message in a scenario as fast as proposed.

7.2.0.3 Movement model

As important as to know the probability of receiving the message is to determine: What are the longer and shorter distances until $c2$ receives the message? Thus, we can find out if it is possible to break in time. Another question is: “What is the expected deceleration for $c2$ to stop?”. Two properties below show respectively these formal specifications:

VI) $R\{\text{"muvCar2"}\}=? [F \text{ send_c2}=1]$

VII) $\text{filter}(\text{max}, \nabla R\{\text{"aCar2"}\}=? [I=t], (\text{pos_l}=70)\&(v_c1=5) \&(v_c2=0)\&(v_l=5)\&\dots\&(\text{time_c1}=\text{Tout}))$

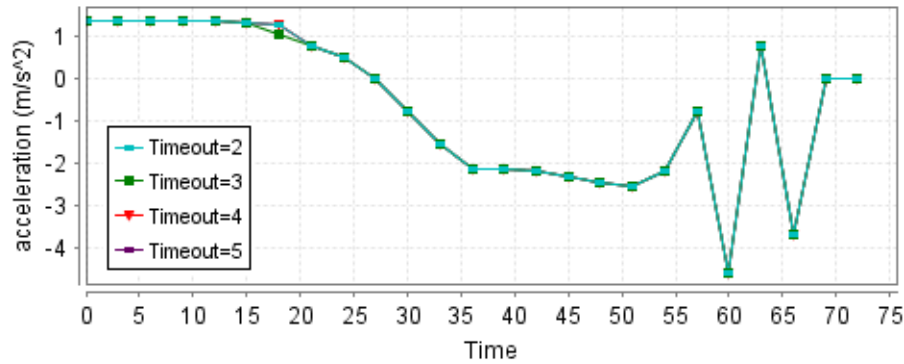
Table 7.1 also shows the results for *Property VI*. These minimum and maximum distance until *c2* to receive the message can be explored using the counterexample provided by PRISM. Therefore, we edited the model to start a braking/deceleration in *c2* when the condition $send_c2=1$ is true. Thus, it is possible to analyze the impact of speed and the distance between current position and the end of road when *c2* receives the warning message. Therefore, it was necessary to implement the deceleration strategy to complement the code presented in Figure 7.2 and change the guard condition to maintain the value 1 to variable $send_c2$, since it was assigned. Both re-implementations were made only for *c2's* modules.

Property VII makes usage of the *filter* command to check specifically the counterexample available by previous property. Thus, we can explore the bounds of distance to different *timeouts*. The filter in this property represents an example of variables that affect the lower limit, such as the initial speeds and leader position. Properties to the upper limits are similar and it was omitted.

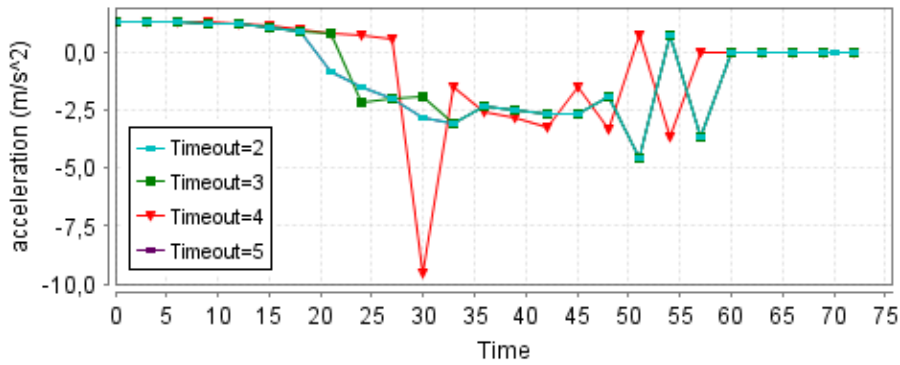
Figure 7.13 shows the results of properties above, it indicates *c2's* acceleration evolution. The first graph features the lower limit, i.e. the *c2* traveled few meters until it receive the warning. Thus, there is larger space and more time to stop. The deceleration is smoother and slower until to steady at zero. Graph 7.13a shows that the acceleration's behavior is not affected by *timeout*, because the warning is received in the beginning. The oscillation at the end is the action modeled by IDM which tries to copy the idea of the stop-and-follow a few meters of the obstacle. The accentuated oscillation is caused by the use of the integer variables in this work.

Graph 7.13b presents the deceleration to the upper limit, i.e. the *c2* receives a message as close as possible to the end. Thus, it will have a lower space and time to stop. Acceleration oscillates more sharply and steady at zero in a shorter time than first one. This is uncomfortable for the vehicle occupants. Besides, the *timeout* set up to 4 presents an abrupt slowdown, while a comfortable braking is about 3 m/s^2 [Wang et al., 2004]. In this scenario, coincidentally, $timeout=5$ had the same behavior set to the value 2. However, for $timeout=3$ the deceleration is more abrupt at the beginning, but still acceptable, after it follows the pattern of the lowest *timeout*.

The results suggest that over large distances, the *timeout* does not affect the deceleration. Surprisingly, for shorter distances the $timeout=5$ results in a more comfortable slowdown than the value 4. However, the probability to receive the message decreases with these values. Using a *timeout* of 2 or 3, the deceleration is comfortable regardless of the distance between sender and receiver, besides there is a higher probability of communicating. Value 3 is better because it generates less traffic than 2 and it has a communication probability of 99%, respected values, against 100%



(a) Larger distance to stop.



(b) Shorter distance to stop.

Figure 7.13: The acceleration behavior in two different scenarios.

to the lowest *timeout*.

7.3 Conclusion

We have presented a formal modeling and analysis of VANET application using Probabilistic Model Checking to represent a Vehicular Warning System. A microscopic vision was proposed to provide a detailed understanding. This was possible using analytical formulas to represent position, speed, acceleration and signal propagation. Some setup were explored, such the time to retransmit the message (*timeout*). The model shows the influence that the timeout, initial speeds and leader position have on communication. Varying from 2 to 100% the chance of message's reception. Furthermore, the study suggests that *timeout* value set up to 3 is a good option over all initial states in the modeled scenario.

This work serves as a guide to verify VANETs in a complete way, addressing motion, network and signal propagation. It is able to extend static network models considering smooth motion and human driving patterns in its mathematical link layer,

considering speed constraints and obstacles.

Regarding to the model's scalability, is hard to predict the number of cars which a given scenario can support. This occurs by the nature of model checking, where the increase of a single variable or a simple change in the variable declaration order may increase significantly the number of states in the model. For the presented scenarios in this work, up to eight vehicles were supported on a computer with 64 Gigabytes of memory. However the amount of cars can be reduced according to the level of detail (number and size of the variables) that the studied protocol requires. A possible solution to this problem is to increase the granularity from microscopic to mesoscopic. Thus, one or more cars are dealt in detail and another vehicle will represent the influence of a vehicles group.

Chapter 8

Conclusions

Outline. This chapter summarizes this research and discusses directions for future study. Section 8.1 presents the conclusion of this thesis. Section 8.2 lists our contributions and the Section 8.3 presents the future work.

8.1 Final Remarks

Vehicular Ad Hoc Networks are an important part of the Intelligent Transportation Systems and perform the integration and communication between sensors, vehicles and fixed equipment on the roadside. VANETs are a special kind of Mobile Ad-hoc Networks, where vehicles with processing power and wireless communications create a spontaneous network along the roads. In this context, industry and academia have been developing standards and prototypes for vehicular networks, which have huge commercial value. Thus, several applications have been proposed, from entertainment to the prevention of accidents. However, these last ones should be executed without error, once they involve human lives, therefore, it is essential to test and analyze them in order to prevent loss of life.

Simulation is widely used to check new protocols and applications. However, there are challenges that must be addressed by the research community like the specifications of APIs for coupling traffic flow and networking simulators, the modeling how drivers react to the additional information provided by VANETs and the use of Real-world measurements to capture the probabilistic effects of small scale fading that have a significant impact on packet reception.

An efficient alternative approach is the Probabilistic Model Checking, which we apply in this research for the verification and analysis of VANETs. Three models with incremental levels of detail were presented sequentially. A model for an intersection

of two urban roads managed by a virtual traffic signal and another model responsible for investigating an overtake situation, both with the focus on vehicular movement. The last analysis has studied a Vehicular Warning System, which shows the influence that the timeout, the initial speeds and the leader position have on communication. All these tests have been built using the PMC tool called PRISM. We noticed that different types of abstractions can be created. Thus, microscopic or macroscopic models can be produced. The analysis has proved to be very useful. They can show the probability that an event occurs and tests involving acceleration, velocity or distance can be extracted.

The conceptual map presented in Figure 3.4, which represents the functionalities of a realistic vehicular mobility model can be created in PMC. **Motion constraints** can be modeled taking into account streets, highways, crossroads, curve roads and other vehicles. **Traffic generator** can be implemented defining different kinds of vehicles (cars, emergency vehicles, trucks) in a macroscopic and in a microscopic way. The **Time** can be parameterized to describe different mobility configurations for a specific time of the day and finally, the impact of a communication protocol or any other source of information on the motion patterns can be implemented synchronizing the modules responsible for movement and network, implementing the **External influences**.

This research has shown that PMC can be used to model and analyze the flow of vehicles managed by VANETs. It extends the traditional use of model checking for verification just from communication protocols (network model) to the analysis of movements patterns (mobility model) in vehicular networks. Thus, PMC can be used to obtain valuable information of VANET in a simple and complete way. In other words, VANET verifications will consider network, signal propagation and movement. This type of analysis can provide a better understanding of how the VANET can influence in urban traffic. Thus, protocols and applications can be easily studied and through model checking advantages, abnormal situations can be identified.

Finally, we have presented our modeling guidelines to supply a consistent and plausible approach to the development of VANETs models in probabilistic model checking. It provides direction on the scope and common approaches to represent protocols/applications. Moreover, a continual evolution of this guidelines is encouraged.

Our guideline provides a sequence of steps where important issues which the researchers should not omit have been defined. During the design phase we have proposed an architecture that includes microscopic aspects to represent movement, network and signal propagation. Thus, our modeling structure will help the researchers to maintain focus on the most essential issues of the problem.

Our architecture cares about low coupling of modules that represents movement, network and signal propagation. However, we can reimplement some of them and use others; this makes future analysis easier, because there will be a know-how about what aspects we must model and a knowledge base could be created. Furthermore, the studies, which obey the same level of specification, can be compared in an easier way. Thus, our guidelines are aligned with our goal to improve the quality of the analysis in VANETs.

We can also conclude that scalability is not a serious problem for our modeling guidelines, since we are proposing small scenarios, for example, intersections or avenues fragment. In addition, critical situations involving few vehicles, for instance, despite the number of vehicles on a highway, in an overtake maneuver, only the faster one, the overtaken vehicle and the first car coming in the opposite direction are important for the modeling. As another example, following the same principle, for the leader election in a virtual traffic sign, only the vehicles ahead are candidates for leadership, so, using until four cars in a lane and only one, in the other direction are enough to generate plausible results.

In addition, our work provides an introduction to VANETs and a critical presentation of the simulators usage in this type of network. The state of the art in virtual traffic lights is also presented and finally, a survey showing the main ideas on how the related works are modeling VANET protocols is depicted.

8.2 Our Contributions

We could briefly listing our main contributions like:

- In contrast to the related works which treat non-determinism to broadcast the messages to static nodes, we take into account traffic flow, computer networks and radio-propagation according to the movement;
- This work contributes to formal analysis in VANETs presenting a modeling structure with a microscopic granularity, which describes the traffic flow in details together with a stochastic way to represent the possibility of receiving a message with a probability p according to each vehicle's position;
- We proposed guidelines for building and verifying vehicular networks using our modeling. Thus, future works may use the suggested instructions to create the models with a similar abstraction level making the studies and results comparable.

They consider important questions like non-determinism to broadcast the messages. However, traffic flow, computer networks and radio-propagation are necessary and rarely explored together in this field. Furthermore,

8.3 Future work

The guidelines and the modeling approach for vehicular network proposed in this thesis must be complemented by the following future work:

- A complete study should be done to determine which probabilistic model checkers are the best to our guidelines. Our work requires some minimum features, such as the ability to quantify states and transitions, however studies regarding processing time and memory spent using our basic modeling structure were not performed. Moreover, a language with more advanced mathematical features will give more expressiveness for the encoding of both signal propagation and movement models.
- Our models' implementation have been using discrete-time Markov chains (DTMCs) or continuous-time Markov chains (CTMCs). However, several other types of probabilistic models could be used, such as probabilistic automaton (PAs) or probabilistic timed automata (PTAs). Thus, a comparative study raising the various arguments in favour of and against these approaches to encode our guidelines could be performed.
- A detailed study regarding to the best analytical method to represent the signal propagation using our guidelines must be carried out. This study must find out the best trade-off between realism and effectiveness, i.e. the communication should be represented in a realistic way and at the same time it has to be implementable in the main model checkers used by academic and industrial community, i.e. the software must support the mathematical functions proposed by the model. In addition, this analytical model can not forbid the spent time to process the analysis.
- A detailed study regarding to the best analytical method to represent the movement of vehicles according to our guidelines should be performed. This study should also find out the best trade-off between realism and the ability to promote the desired results, that is, it has to be implementable in the main probabilistic model checkers and it cannot spend much time during the analysis processing.

- We have realized that it is easy to adapt the mobility model to various situations. For instance, to implement a curve road, it is necessary to just change the desired speed to a lower value than the used on a straight road, so the vehicles will reduce the speed. Thus, we can abstract, for a certain distance, that the vehicles consider this slower speed while crossing the curve. However, through these type of abstractions, many short traffic scenarios could be implemented and made available to the community. Furthermore, could be created properties templates to explore each scenario. These basic queries should be well documented to guide initial researchers, the idea is to encourage changes and increases in a tutorial format.
- Creating an integrated development environment (IDE) for encoding models in the vehicular network analysis, among several aspects: the software should intuitively allow, through graphic user interface, the choice of different scenarios, signal propagation and mobility models, besides the number and types of vehicles. Thus, the tool could generate the models following our proposed structure, so that, the researcher could focus only on the representation of the protocol/application under study. This software should also accept the inclusion of new scenarios and analytical formulas. Furthermore, the software could facilitate the creation and maintenance of a database about model checking in VANETs. This action would help to make more popular the usage of this formal method and ensure the correct operation of the proposed applications/protocols in vehicular networks.

Bibliography

- Acosta-Marum, G. and Ingram, M. A. (2006). Model development for the wideband expressway vehicle-to-vehicle 2.4 ghz channel. In *WCNC*, pages 1283–1288. IEEE.
- Adobe Resource Center (2016). Naming conventions. http://help.adobe.com/en_US/AS2LCR/Flash_10.0/help.html. Accessed: 3 mar. 2016.
- Akcelik, R., B. M. and Chung (1998). An evaluation of scats master isolated control.
- Alves, R. and et al. (2009). Redes veiculares: Princípios, aplicações e desafios. *in Minicursos do Simpósio Brasileiro de Redes de Computadores - SBRC'2009*.
- Argent-Katwala, A., Clark, A., Foster, H., Gilmore, S., Mayer, P., and Tribastone, M. (2008). Safety and response-time analysis of an automotive accident assistance service. In *Leveraging Applications of Formal Methods, Verification and Validation, Third International Symposium, ISO LA 2008, Porto Sani, Greece, October 13-15, 2008. Proceedings*, pages 191--205.
- Bai, F., Krishnan, H., Sadekar, V., Holland, G., and Elbatt, T. (2006). Towards Characterizing and Classifying Communication-based Automotive Applications from a Wireless Networking Perspective. In *In Proceedings of IEEE Workshop on Automotive Networking and Applications (AutoNet)*.
- Baier, C., Haverkort, B. R., Hermanns, H., and Katoen, J.-P. (2010). Performance evaluation and model checking join forces. *Commun. ACM*, 53(9):76–85.
- Barnett, B., Richardson, S., Evans, R., and Peeters, L. (2012). Australian groundwater modelling guidelines. Technical report, National Water Commission.
- Bengtsson, J., Larsen, K., Larsson, F., Pettersson, P., and Yi, W. (1995). Uppaal - a tool suite for automatic verification of real-time systems. In *Hybrid Systems III, LNCS 1066*, pages 232--243. Springer-Verlag.

- Berezin, S., Campos, S. V. A., and Clarke, E. M. (1998). Compositional reasoning in model checking. In *Revised Lectures from the International Symposium on Compositionality: The Significant Difference*, COMPOS'97, pages 81--102, London, UK, UK. Springer-Verlag.
- Berhmann, G., David, A., Håkansson, J., Hendriks, M., Larsen, K. G., Pettersson, P., and Yi, W. (2006). Uppaal 4.0. In *3rd International Conference on Quantitative Evaluation of Systems (QEST'06)*, pages 125--126. IEEE Computer Society.
- Blum, J., Eskandarian, A., and Hoffman, L. (2004). Challenges of intervehicle ad hoc networks. *Intelligent Transportation Systems, IEEE Transactions on*, 5(4):347--351. ISSN 1524-9050.
- Boban, M. and Vinhoza, T. T. V. (2011). Modeling and simulation of vehicular networks: Towards realistic and efficient models. In *Mobile Ad-Hoc: Applications*. Intech.
- Bollig, B. and Wegener, I. (1996). Improving the variable ordering of OBDDs is NP-complete. *Computers, IEEE Transactions on*, 45(9):993--1002. ISSN 0018-9340.
- Bononi, L., Di Felice, M., Bertini, M., and Croci, E. (2006). Parallel and distributed simulation of wireless vehicular ad hoc networks. In *Proc. of the 9th ACM int. symposium on Modeling analysis and simulation of wireless*, pages 28--35, New York, USA.
- Bouassida, M. S. and Shawky, M. (2010). A cooperative congestion control approach within vanets: formal verification and performance evaluation. *EURASIP J. Wirel. Commun. Netw.*, 2010:11:1--11:12. ISSN 1687-1472.
- Boulis, A. (2013). Castalia: A simulator for wireless sensor networks. <http://castalia.research.nicta.com.au/index.php/en/>. Access date: 06 nov. 2013.
- Boulis, A., Fehnker, A., Fruth, M., and McIver, A. (2008). Cavi – simulation and model checking for wireless sensor networks. QEST.
- Braz, F., Amaral, J., Ferreira, B., Cruz, J., Faria-Campos, A., and Campos, S. (2013a). A probabilistic model checking analysis of the potassium reactions with the palytoxin and Na^+/K^+ -ATPase complex. In Setubal, J. and Almeida, N., editors, *Advances in Bioinformatics and Computational Biology*, volume 8213 of *Lecture Notes in Computer Science*, pages 181--193. Springer International Publishing.

- Braz, F. A. F., Amaral, J., Ferreira, B., Cruz, J. S., Faria-Campos, A. C., and Campos, S. V. A. (2013b). A probabilistic model checking analysis of the potassium reactions with the palytoxin and $na + /k + -atpase$ complex. In *Advances in Bioinformatics and Computational Biology - 8th Brazilian Symposium on Bioinformatics, BSB 2013, Recife, Brazil, November 3-7, 2013, Proceedings*, pages 181--193.
- Bryant, R. E. (1986). Graph-based algorithms for boolean function manipulation. *IEEE Trans. Comput.*, 35:677--691. ISSN 0018-9340.
- Bureau, U. C. (2013). The Tiger/Line Database. <http://www.census.gov/geo/www/tiger/>. Access date: 23 dec. 2012.
- C. Dehnert, J.-P. K. and Parker, D. (2013). SMT-based bisimulation minimisation of Markov models. In Giacobazzi, R., Berdine, J., and Mastroeni, I., editors, *Proc. 14th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'13)*, volume 7737 of *LNCS*. Springer. To appear.
- C2C-CC (2007). Car to Car Communication Consortium Manifesto: Overview Overview of the C2C-CC System. Tech Rep. Technical report, Car to Car Communication Consortium.
- C2C-CC (EU) (2013). Car-to-car communication consortium. <http://www.car-to-car.org/>. Access date: 25 out. 2013.
- CAMP-US (2013). Collision Avoidance Metrics Partnership-CAMP (US). <http://www.its.dot.gov/cicas/>. Access date: 2 out. 2013.
- Choffnes, D. R. and Bustamante, F. E. (2005). An integrated mobility and traffic model for vehicular wireless networks. In Laberteaux, K. P., Hartenstein, H., Johnson, D. B., and Sengupta, R., editors, *Vehicular Ad Hoc Networks*, pages 69--78. ACM.
- Chou, L., Tseng, J., and Yang, J. (2013). Adaptive virtual traffic light based on vanets for mitigating congestion in smart city. In *The Third International Conference on Digital Information and Communication Technology and its Applications*, pages 40--44.
- Christian, A. (2009). Reliable model checking for wsns. In *Proc. of the 8th GI/ITG KuVS Fachgesprach*.
- Cimatti, A., Clarke, E., Giunchiglia, F., and Roveri, M. (2000). Nusmv: a new symbolic model checker. *International Journal on Software Tools for Technology Transfer*, 2:2000.

- Clarke, E. and Emerson, E. (1982). Design and synthesis of synchronization skeletons using branching time temporal logic. In Kozen, D., editor, *Logics of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer Berlin / Heidelberg. 10.1007/BFb0025774.
- Clarke, E., Emerson, E., Jha, S., and Sistla, A. (1998). Symmetry reductions in model checking. In Hu, A. and Vardi, M., editors, *Computer Aided Verification*, volume 1427 of *Lecture Notes in Computer Science*, pages 147–158. Springer Berlin Heidelberg.
- Clarke, E., Grumberg, O., and Peled, D. (1999). *Model Checking*. MIT Press.
- Clarke, E. M., Emerson, E. A., and Sistla, A. P. (1986). Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.*, 8:244–263. ISSN 0164-0925.
- Clarke, E. M., Faeder, J. R., Langmead, C. J., Harris, L. A., Jha, S. K., and Legay, A. (2008). Statistical model checking in BioLab: Applications to the automated analysis of T-Cell receptor signaling pathway. In *Proceedings of the 6th International Conference on Computational Methods in Systems Biology, CMSB '08*, pages 231–250, Berlin, Heidelberg. Springer-Verlag.
- Conceição, H., Damas, L., Ferreira, M., and Barros, J. a. (2008). Large-scale simulation of v2v environments. In *Proceedings of the 2008 ACM symposium on Applied computing, SAC '08*, pages 28–33, New York, NY, USA. ACM.
- Correia, S. L. O. B. (2011). *Roteamento em Redes Veiculares Utilizando Colônias de Formigas e Predição de Mobilidade*. PhD thesis, Universidade Estadual do Ceará.
- Crepalde, M. (2011). Modelagem e análise de sistemas de transporte de Íons em membranas celulares usando verificação de modelos. Master’s thesis, Universidade Federal de Minas Gerais.
- Dashtinezhad, S., Nadeem, T., Dorohonceanu, B., and Borcea, C. (2004). Trafficview: A driver assistant device for traffic monitoring based on car-to-car communication. In *In IEEE Semiannual Vehicular Technology*, pages 17–19.
- Davies, J. J., Beresford, A. R., and Hopper, A. (2006). Scalable, distributed, real-time map generation. *IEEE Pervasive Computing*, 5(4):47–54. ISSN 1536-1268.
- Dirk Helbing, T. and Martin, A. K. (2007). General lane-changing model mobil for car-following models. In *Transportation Research Record: Journal of the*

- Transportation Research Board*, pages 86–94. Transportation Research Board of the National Academies.
- DoT, U. (2004). Toolbox of Countermeasures and Their Potential Effectiveness to Make Intersections Safer - Issue Brief 8. Technical report, Inst. Transportation Eng.
- DoT, U. (2007). Traffic Signals - Issue Brief 5. Technical report, Inst. Transportation Eng.
- Edwards, J. (2001). Process algebras for protocol validation and analysis.
- Emerson, E. A. and Clarke, E. M. (1980). Characterizing correctness properties of parallel programs using fixpoints. In *Proceedings of the 7th Colloquium on Automata, Languages and Programming*, pages 169–181, London, UK. Springer-Verlag.
- Engelstad, Paal E. and Osterb, Olav N. (2005). Non-saturation and saturation analysis of iee 802.11e edca with starvation prediction. In *Proceedings of the 8th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems, MSWiM '05*, pages 224–233, New York, NY, USA. ACM.
- EPEC (2015). The Guide to Guidance. Technical report, European PPP Expertise Centre.
- ETSI (2011). Intelligent Transport Systems (ITS): Vehicular Communications, Basic Set of Applications. Tech. Rep. Technical report, European Telecommunications Standards Institute.
- Fehnker, A., Fruth, M., and McIver, A. (2009). Graphical modelling for simulation and formal analysis of wireless network protocols. In Butler, M., Jones, C. B., Romanovsky, A., and Troubitsyna, E., editors, *Methods, Models and Tools for Fault Tolerance*, volume 5454 of *Lecture Notes in Computer Science*, pages 1–24. Springer.
- Fehnker, A., van Hoesel, L., and Mader, A. (2007). Modelling and verification of the lmac protocol for wireless sensor networks.
- Ferreira, B., Braz, F., and Campos, S. (2012a). VANET model. www.dcc.ufmg.br/~bruno.ferreira/sbmf2012/. Access date: 23 dec. 2014.
- Ferreira, B., Braz, F., and Campos, S. (2014a). VANET movement model. www.dcc.ufmg.br/~bruno.ferreira/sbrc2015/. Access date: 23 dec. 2014.

- Ferreira, B., Braz, F., Loureiro, A., and Campos, S. (2015a). A probabilistic model checking analysis of vehicular ad-hoc networks. In *Vehicular Technology Conference (VTC Spring), 2015 IEEE 81st*, pages 1–7.
- Ferreira, B., Braz, F. A. F., and Campos, S. V. A. (2012b). A probabilistic model checking approach to investigate vehicular networks. In *Proc. 15th Brazilian Symposium of Formal Methods*.
- Ferreira, B., Braz, F. A. F., and Campos, S. V. A. (2014b). A probabilistic model checking analysis of a realistic vehicular networks mobility model. In *Proc. 17th Brazilian Symposium of Formal Methods*.
- Ferreira, B., Cunha, F., Braz, F., Mini, R., Loureiro, A., and Campos, S. (2015b). Intelligent service to perform overtaking in vehicular networks. In *ISCC 2015, The 20th IEEE Symposium on Computers and Communications*, pages 552–559.
- Ferreira, M., Conceição, H., Fernandes, R., and Tonguz, O. K. (2009). Stereoscopic aerial photography: an alternative to model-based urban mobility approaches. In Shorey, R., Weimerskirch, A., Jiang, D., and Mauve, M., editors, *Vehicular Ad Hoc Networks*, pages 53–62. ACM.
- Ferreira, M. and d’Orey, P. M. (2012). On the impact of virtual traffic lights on carbon emissions mitigation. *IEEE Transactions on Intelligent Transportation Systems*, 13(1):284–295.
- Ferreira, M., Fernandes, R., Conceição, H., Viriyasitavat, W., and Tonguz, O. K. (2010). Self-organized traffic control. VANET ’10, New York, NY, USA. ACM.
- Fourneau, J., Kloul, L., and Valois, F. (2002). Performance modelling of hierarchical cellular networks using PEPA. *Perform. Eval.*, 50(2/3):83–99.
- Furukawa, Y. (2005). Overview R&D on Active Safety in Japan. In *ITS Europe*.
- Giordano, E., Frank, R., Pau, G., and Gerla, M. (2010). Corner: a realistic urban propagation model for vanet. In *Proceedings of the 7th international conference on Wireless on-demand network systems and services, WONS’10*, pages 57–60, Piscataway, NJ, USA. IEEE Press.
- Gipps, P. G. (1986). A model for the structure of lane-changing decisions. *Transportation Research Part B: Methodological*, 20(5):403–414.

- Godefroid, P., Peled, D., and Staskauskas, M. (1996). Using partial-order methods in the formal validation of industrial concurrent programs. *Software Engineering, IEEE Transactions on*, 22(7):496–507. ISSN 0098-5589.
- Goguen, J. A. (1984). Parameterized programming. *IEEE Trans. Software Eng.*, 10(5):528–544.
- Gradinescu, V., Gorgorin, C., Diaconescu, R., Cristea, V., and Iftode, L. (2007). Adaptive traffic lights using car-to-car communication. In *VTC Spring*, pages 21–25. IEEE.
- Harri, J., Filali, F., and Bonnet, C. (2009). Mobility models for vehicular ad hoc networks: a survey and taxonomy. *IEEE Communications Surveys & Tutorials*, 11. ISSN 1553-877X.
- Hartenstein, H., Laberteaux, K., and Ebrary, I. (2010). *VANET: vehicular applications and inter-networking technologies*. Wiley Online Library.
- Hartenstein, H. and Laberteaux, K. P. (2008). A tutorial survey on vehicular ad hoc networks. *Comm. Mag.*, 46(6):164–171. ISSN 0163-6804.
- Hassnaa Moustafa, Sidi Mohammed Senouci, M. J. (2009). Introduction to vehicular networks. In *Vehicular Networks Techniques, Standards, and Applications*. Auerbach Publications.
- Helbing, D. (2001). Traffic and related self-driven many-particle systems. *Rev. Mod. Phys.*, 73:1067–1141.
- Herald Wheels (2013). V2V tech driving forward. <http://thechronicleherald.ca/wheelsnews/1158589-v2v-tech-driving-forward>. Access date: 8 nov. 2013.
- Hermanns, H. and Katoen, J.-P. (2001). *CONCUR 2001 — Concurrency Theory: 12th International Conference Aalborg, Denmark, August 20–25, 2001 Proceedings*, chapter Performance Evaluation:= (Process Algebra + Model Checking) X Markov Chains, pages 59–81. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Hillston, J. (1996). *A Compositional Approach to Performance Modelling*. Cambridge University Press, New York, NY, USA. ISBN 0-521-57189-8.
- Hillston, J. (2005). Tuning systems: From composition to performance (the needham lecture). *Comput. J.*, 48(4):385–400.

- Hillston, J. and Ribaudó, M. (2004). Modelling mobility with PEPA nets. In *Computer and Information Sciences - ISCIS 2004, 19th International Symposium, Kemer-Antalya, Turkey, October 27-29, 2004. Proceedings*, pages 513–522.
- Ho, I. W. H., Leung, K. K., Polak, J. W., and Mangharam, R. (2007). Node connectivity in vehicular ad hoc networks with structured mobility. In *Proc. of the 32nd IEEE Conference on Local Computer Networks*, pages 635–642, Washington, USA.
- Hoogendoorn, S. P. and Bovy, P. H. L. (2001). State-of-the-art of vehicular traffic flow modelling. In *Delft University of Technology, Delft, The*, pages 283–303.
- Jahanian, M. H., Amin, F., and Jahangir, A. H. (2015). Analysis of tesla protocol in vehicular ad hoc networks using timed colored petri nets. In *Information and Communication Systems (ICICS), 2015 6th International Conference on*, pages 222–227.
- Jain, R. (1991). *The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling*. Wiley professional computing. Wiley. ISBN 978-0-471-50336-1.
- Jang, B. and Sichitiu, M. L. (2012). Ieee 802.11 saturation throughput analysis in the presence of hidden terminals. *IEEE/ACM Trans. Netw.*, 20(2):557–570. ISSN 1063-6692.
- Karagiannis, G., Altintas, O., Ekici, E., Heijenk, G., Jarupan, B., Lin, K., and Weil, T. (2011). Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions. *IEEE Communications Surveys & Tutorials*.
- Kesting, A., Treiber, M., and Helbing, D. (2010). Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity. *Royal Society of London Philosophical Transactions Series A*, 368:4585–4605.
- Khosroshahy, M. (2007). IEEE 802.11 and propagation modeling: A survey and a practical design approach.
- Killat, M. and Hartenstein, H. (2009). An empirical model for probability of packet reception in vehicular ad hoc networks. *EURASIP J. Wirel. Commun. Netw.*, 2009:4:1--4:12. ISSN 1687-1472.
- Konur, S. and Fisher, M. (2011). Formal analysis of a vanet congestion control protocol through probabilistic verification. In *VTC Spring*, pages 1–5. IEEE.

- Kwiatkowska, M. and Norman, G. (2011). PRISM - Property Specification. <http://www.prismmodelchecker.org/manual/PropertySpecification/Reward-basedProperties>. Access date: 28 jan. 2014.
- Kwiatkowska, M., Norman, G., and Parker, D. (2004). Probabilistic symbolic model checking with PRISM: A hybrid approach. *International Journal on Software Tools for Technology Transfer (STTT)*, 6(2):128--142.
- Kwiatkowska, M., Norman, G., and Parker, D. (2007). Stochastic model checking. In Bernardo, M. and Hillston, J., editors, *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM'07)*, volume 4486 of *LNCS (Tutorial Volume)*, pages 220--270. Springer.
- Kwiatkowska, M., Norman, G., and Parker, D. (2008). Using probabilistic model checking in systems biology. *SIGMETRICS Perform. Eval. Rev.*, 35(4):14--21. ISSN 0163-5999.
- Kwiatkowska, M., Norman, G., and Parker, D. (2009). PRISM: Probabilistic model checking for performance and reliability analysis. *SIGMETRICS Perform. Eval. Rev.*, 36(4):40--45. ISSN 0163-5999.
- Kwiatkowska, M., Norman, G., and Parker, D. (2011). PRISM 4.0: Verification of probabilistic real-time systems. In *Proc. CAV*. Springer.
- Leon-Garcia, A. (2008). *Probability, Statistics, and Random Processes for Electrical Engineering*. Pearson/Prentice Hall, Upper Saddle River, NJ, third edition. ISBN 9780131471221 0131471228.
- Levins, R. (1966). The Strategy of Model Building in Population Biology. *American Scientist*, 54:421--431.
- Lighthill, M. J. and Whitham, G. B. (1955). On kinetic waves: Ii) a theory of traffic flow on long crowded roads. In *Proc. Royal Society A229*, pages 281--345.
- Lin, W.-Y., Li, M.-W., Lan, K.-C., and Hsu, C.-H. (2012). A comparison of 802.11a and 802.11p for v-to-i communication: A measurement study. In Zhang, X. and Qiao, D., editors, *Quality, Reliability, Security and Robustness in Heterogeneous Networks*, volume 74 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 559--570. Springer Berlin Heidelberg.

- Lomuscio, A., Strulo, B., Walker, N. G., and Wu, P. (2010). Model checking optimisation based congestion control algorithms. *Fundam. Inf.*, 102(1). ISSN 0169-2968.
- Malone, D., Duffy, K., and Leith, D. (2007). Modeling the 802.11 distributed coordination function in nonsaturated heterogeneous conditions. *IEEE/ACM Trans. Netw.*, 15(1):159--172. ISSN 1063-6692.
- Mangel, T., Klemp, O., and Hartenstein, H. (2011). 5.9 ghz inter-vehicle communication at intersections: a validated non-line-of-sight path-loss and fading model. *EURASIP J. Wireless Comm. and Networking*, 2011:182.
- Mangharam, R., Weller, D. S., Stancil, D. D., Rajkumar, R., and Parikh, J. S. (2005). Groovesim: a topography-accurate simulator for geographic routing in vehicular networks. In Laberteaux, K. P., Hartenstein, H., Johnson, D. B., and Sengupta, R., editors, *Vehicular Ad Hoc Networks*, pages 59--68. ACM.
- Martinez, F. J., Toh, C. K., Cano, J.-C., Calafate, C. T., and Manzoni, P. (2011). A survey and comparative study of simulators for vehicular ad hoc networks (vanets). *Wirel. Commun. Mob. Comput.*, 11(7):813--828. ISSN 1530-8669.
- Maurer, J., Fugen, T., Schafer, T., and Wiesbeck, W. (2004). A new inter-vehicle communications (ivc) channel model - vtc2004-fall. In *Vehicular Technology Conference*, pages 9--13.
- McMillan, K. L. (1992). The smv system.
- McMillan, K. L. (1992). *Symbolic model checking: an approach to the state explosion problem*. PhD thesis, Carnegie Mellon University.
- Nagel, K. and Schreckenberg, M. (1992). A cellular automaton model for freeway traffic. *Journal de Physique I*, 2(12):2221--2229. ISSN 1155-4304.
- Nagel, R. and Eichler, S. (2008). Efficient and realistic mobility and channel modeling for vanet scenarios using omnet++ and inet-framework. In *OMNeT++ 2008: Proceedings of the 1st International Workshop on OMNeT++ (hosted by SIMUTools 2008)*, ICST, Brussels, Belgium, Belgium. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
- Naik, V. G. and Sistla, A. P. (1994). Modeling and verification of a real life protocol using symbolic model checking. In Dill, D. L., editor, *Computer Aided Verification*,

- 6th International Conference, CAV 94, Stanford, California, USA, June 21-23, 1994, Proceedings*, volume 818 of *Lecture Notes in Computer Science*, pages 194–206. Springer.
- Nakagami, M. (1960). The m-distribution – A general formula of intensity distribution of rapid fading. In Hoffmann, W. C., editor, *Statistical Methods in Radio Wave Propagation*. Elmsford, NY.
- NASVA-JP (2013). Advanced safety vehicle (asv) program. http://www.nasva.go.jp/mamoru/en/assessment_car/asv.html. Access date: 25 out. 2013.
- Nekovee, M. (2005). Sensor networks on the road: the promises and challenges of vehicular ad hoc networks and vehicular grids. In *In Proc. of the Workshop on Ubiquitous Computing and e-Research*.
- Neudecker, T., An, N., Tonguz, O. K., Gaugel, T., and Mittag, J. (2012). Feasibility of virtual traffic lights in non-line-of-sight environments. In *Proceedings of the ninth ACM international workshop on Vehicular inter-networking, systems, and applications*, VANET '12, pages 103–106, New York, NY, USA. ACM.
- Neves, F., Cardote, A., Moreira, R., and Sargento, S. (2011). Real-world evaluation of iee 802.11p for vehicular networks. In *Proceedings of the Eighth ACM International Workshop on Vehicular Inter-networking*, VANET '11, pages 89–90, New York, NY, USA. ACM.
- Obiniyi, A. A., Soroyewun, M. B., and Abur, M. M. (2014). Article: New innovations in performance analysis of computer networks: A review. *International Journal of Applied Information Systems*, 6(8):1–10. Published by Foundation of Computer Science, New York, USA.
- OMB (2007). Final Bulletin for Agency Good Guidance Practices. Technical report, EUA Executive Office.
- Othman, J. B., Mokdad, L., Cheikh, M. O., and Sene, M. (2009). Performance analysis of composite web services using stochastic automata networks over ip network. In *Computers and Communications, 2009. ISCC 2009. IEEE Symposium on*, pages 92–97. ISSN 1530-1346.
- Parker, D. (2002). *Implementation of Symbolic Model Checking for Probabilistic Systems*. PhD thesis, University of Birmingham.
- Paromtchik, I. and Laugier, C. (2007). The advanced safety vehicle programme.

- Patrushev, A. (2008). Shortest path search for real road networks and dynamic costs with pgrouting.
- Piorowski, M., Raya, M., Lugo, A. L., Papadimitratos, P., Grossglauser, M., and Hubaux, J.-P. (2008). Trans: realistic joint traffic and network simulator for vanets. *Mobile Computing and Communications Review*, 12(1):31–33.
- Plateau, B. and Atif, K. (1997). Stochastic automata network of modeling parallel systems. *IEEE Transactions on Software Engineering*, 17(10):1093–1108. ISSN 0098-5589.
- Plateau, B. and Stewart, W. J. (1997). Stochastic automata networks. In *Computational Probability*, pages 113–152. Kluwer Academic Press.
- Prj, P. (2008). Detailed description of selected use cases and corresponding technical requirements. Tech. Rep. Technical report, PreDrive C2X project - Preparing an European FOT on Car-2-X Communication.
- Puigjaner, R. (2003). Performance modelling of computer networks. In *Proceedings of the 2003 IFIP/ACM Latin America Conference on Towards a Latin American Agenda for Network Research*, LANC '03, pages 106–123, New York, NY, USA. ACM.
- Queille, J. P. and Sifakis, J. (1982). A temporal logic to deal with fairness in transition systems. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, SFCS '82, pages 217–225, Washington, DC, USA. IEEE Computer Society.
- Robertson, D. I. and Bretherton, R. D. (1991). Optimizing networks of traffic signals in real time-the SCOOT method. *IEEE Transactions on Vehicular Technology*, 40(1):11–15. ISSN 0018-9545.
- Rothery, W. (1992). Car following models, in trac flow theory.
- Rubio, L., Reig, J., and Cardona, N. (2007). Evaluation of nakagami fading behaviour based on measurements in urban scenarios - letter. *AEUE - International Journal of Electronics and Communications*, 61(2):135–138.
- Santos, A., Conceição, H., Mendes, H., and Jordão, N. (2010). *Distributed Virtual Traffic Light System*. PhD thesis, Carnegie Mellon University.
- Sheng, A., Byung-Hyug, L., and Dong-Ryeol, S. (2011). A survey of intelligent transportation systems. In *Computational Intelligence, Communication Systems and Networks (CICSyN), 2011 Third International Conference on*, pages 332–337.

- Song, M. A. J. (2004). *The UML-CAFE: an Environment to Specify and Verify Transactional Systems*. PhD thesis, Universidade Federal de Minas Gerais.
- Strunk, E., Aiello, A., and Knight, J. (2006). A Survey of Tools for Model Checking and Model-Based Development. Technical report, University of Virginia.
- Taliwal, V., Jiang, D., Mangold, H., Chen, C., and Sengupta, R. (2004). Empirical determination of channel characteristics for dsrc vehicle-to-vehicle communication. In *Proceedings of the 1st ACM International Workshop on Vehicular Ad Hoc Networks, VANET '04*, pages 88--88, New York, NY, USA. ACM.
- Teixeira, F. A., e Silva, V. F., Leoni, J. L., Macedo, D. F., and Nogueira, J. M. (2014). Vehicular networks using the {IEEE} 802.11p standard: An experimental analysis. *Vehicular Communications*, 1(2):91 – 96. ISSN 2214-2096.
- Tonguz, O. (2011). Notice of violation of ieee publication principles biologically inspired solutions to fundamental transportation problems. *Communications Magazine, IEEE*, 49(11):106–115. ISSN 0163-6804.
- Torrent-Moreno, M., Santi, P., and Hartenstein, H. (2005). Fair sharing of bandwidth in vanets. In *Proceedings of the 2nd ACM international workshop on Vehicular ad hoc networks, VANET '05*, pages 49--58, New York, NY, USA. ACM.
- Toyota (2012). Sensors and Actuators. Technical report, Toyota Motor Sales.
- Treiber, M., Hennecke, A., and Helbing, D. (2000). Congested traffic states in empirical observations and microscopic simulations. *Rev. E* 62, Issue, 62:2000.
- Van Eenennaam, E. (2008). A survey of propagation models used in vehicular ad hoc network (vanet) research. *Paper written for course Mobile Radio Communication, University of Twente*.
- Vasilakos, A. V., Zhang, Y., and Spyropoulos, T. (2011). *Delay Tolerant Networks: Protocols and Applications*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition. ISBN 1439811083, 9781439811085.
- Vehicle Safety Communications Consortium (2004). *Identify Intelligent Vehicle Safety Applications Enabled by DSRC*.
- Veysey, M. and Moran, G. (2013). Traffic Modelling Guidelines. Technical report, Traffic & Safety Management Group.

- VGrid (2009). Projects - vmesh/vgrid. <http://www.ece.ucdavis.edu/rubinet/projects/vgrid.html>. Access date: 26 nov. 2012.
- VII-US (2012). The vehicle infrastructure integration (vii). <http://www.its.dot.gov/vii/>. Access date: 22 out. 2013.
- VSC-US (2011). Vehicle safety communications projects. www.nhtsa.gov/DOT/NHTSA/NVS/CrashAvoidance/TechnicalPublications/2011/811492A.pdf. Access date: 22 out. 2013.
- Wang, J., Dixon, K. K., Li, H., and Ogle, J. (2004). Normal acceleration behavior of passenger vehicles starting from rest at all-way stop-controlled intersections. In *Traffic Flow Theory and Highway Capacity and Quality of Service 2004*, Transportation Research Record, pages 158–166. Transportation Research Board of the National Academies.
- Wang, S. and Chou, C. (2008). *NCTUns Simulator for Wireless Vehicular Ad Hoc Network Research*. Nova Science Publishers.
- Wang, S.-Y. and Lin, C.-C. (2008). Nctuns 5.0: A network simulator for ieee 802.11(p) and 1609 wireless vehicular network researches. In *VTC Fall*, pages 1–2. IEEE.
- Wei, D. X., Jin, C., Low, S. H., and Hegde, S. (2006). Fast tcp: motivation, architecture, algorithms, performance. *IEEE/ACM Trans. Netw.*, 14(6):1246–1259. ISSN 1063-6692.
- Wu, H., Lee, J., Hunter, M., Fujimoto, R., Guensler, R., and Ko, J. (2005). Efficiency of simulated vehicle-to-vehicle message propagation in atlanta, i-75 corridor. *Transportation Research Record: Journal of the Transportation Research Board*, pages 82–89.
- Younes, H. (2005). Ymer: A statistical model checker. In Etessami, K. and Rajamani, S., editors, *Computer Aided Verification*, volume 3576 of *Lecture Notes in Computer Science*, pages 171–179. Springer Berlin / Heidelberg. 10.1007/11513988_43.
- Yousefi, S., Altman, E., El-Azouzi, R., and Fathy, M. (2008). Analytical model for connectivity in vehicular ad hoc networks. *IEEE Transactions on Vehicular Technology*, 57(6):3341–3356. ISSN 0018-9545.
- Zuniga, M. and Krishnamachari, B. (2004). Analyzing the transitional region in low power wireless links. In *Sensor and Ad Hoc Communications and Networks, 2004*.

IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on, pages 517–526.