Formal Analysis of a VANET Congestion Control Protocol through Probabilistic Verification

Savas Konur and Michael Fisher
Department of Computer Science, University of Liverpool, Liverpool, UK
Email: {Konur, MFisher}@liverpool.ac.uk

Abstract—**Vehicular ad hoc networks (VANETs), which are a class of Mobile ad hoc networks, have recently been developed as a standard means of communication among moving vehicles. Since VANETs are vital to the safety of the vehicles, the infrastructure, and the humans involved, a deep analysis of their potential behaviours is clearly required. In this paper we provide this analysis through the use of formal verification. Specifically, we formally analyse a specific congestion control protocol for VANETs using a probabilistic model checking technique, and investigate its correctness and effectiveness.**

I. Introduction

A *congestion control protocol* is an algorithm which is used to share available resources among nodes within a network [11]. If the available resources are limited, and the network topology and node density change over time, a *fair* sharing of resources becomes difficult. Applying conventional congestion control protocols to *vehicular ad hoc networks* (VANETs) can also be problematic, particularly if we require an efficient protocol that is also able to guarantee reliable and safe communication. This has led to a range of recent studies which have focused on developing new congestion protocols more suitable to the challenges of VANETs: dynamic (and fast) network topology changes; dynamic network density changes; network scale problems; peculiar interference issues; limited bandwidth; etc. Indeed, improvements and refinements to VANET-specific congestion control protocols continue to be made [15], [14], [3], [2]. 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 behaviours, and so protocols analyzed in this way can have unpredictable behaviour due to an incomplete system analysis. Since a VANET can transmit both *safety* and *emergency* messages, then a more reliable method of analysis is clearly essential.

Within Computer Science, a typical solution to this problem is to use *formal verification* techniques to carry out *exhaustive* analysis rather than examining systems through simulation or testing. One particularly well-known and successful form of formal verification is termed *model-checking* [5]. Although model-checking has been used extensively for network protocols [9], it has been rarely used in the area of congestion control. To the best of our knowledge there are only two studies which focus on model checking for the formal analysis of such control problems, as follows. In [2], the suggestion is to verify a congestion control method via the UPPAAL [1] system. However, although this suggestion was made, verification apparently did not succeed due to memory limitations. In [13] a model-checking alternative to the use of "optimisation based approaches" is proposed. The authors use the NuSMV model-checking tool [4] to evaluate a congestion control approach. However, neither of the above formal approaches address some of the more challenging characteristics typical of VANETs such as uncertainty and non-determinism.

In this paper we formally analyse and verify the specific congestion control approach for vehicular ad hoc networks introduced in [2]. (We selected [2] because the proposed protocol is completely dedicated to VANETs.) We then apply a *probabilistic* method to investigate the efficiency of the protocol, analyzing the protocol with the probabilistic model checker PRISM [8]. From the verification results we observe that, by making certain changes, we can improve both the channel utilization and message transmission. Specifically, we show that the modifications we propose allow us achieve a better (i.e. lower) loss rate and delay for safety messages. Our study therefore shows that verification techniques can be very useful in assessing the efficiency and correctness of a congestion protocol, and that the results obtained from such assessments may be used to improve the VANET performance.

The structure of the paper is as follows. In Section II, we present the congestion protocol we aim to analyze. In Section III, we briefly describe probabilistic model checking, and the PRISM tool in particular. In Section IV, we formally model the congestion protocol from Section II and, in Section V, we analyze this model and investigate various refinements brought to light through the verification process. Finally, in Section VI we provide concluding remarks.

II. A Congestion Control Protocol for VANETs

Vehicular ad hoc networks (VANETs) are a standard means of communication, allowing vehicles to communicate with each other, even in the absence of a communication infrastructure, such as that provided by roadside base stations. Due to the complex and dynamic nature of traffic, VANET networks have potentially more challenging characteristics than MANET networks including fast topology changes, dynamic network density, network scale in certain areas (e.g. city centres), interference issues, limited bandwidth etc. [16].

In [2], a congestion control approach based on "dynamic scheduling" and "transmission of priority-based messages" is proposed whereby 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. This protocol has three stages:

a) 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 next.

b) Message scheduling: Based on the priorities assigned, messages are sent to an appropriate channel. In a VANET, packets are accessed through a shared medium [14]. There are some recent standards for the use of this medium. For example, IEEE 802.11p [10] proposes a partition of the bandwidth into several channels: one *control channel (CCH)* and six *service channels (SCHs)*. The control channel is used for *event-driven emergency* messages and *periodic* safety messages [14], and the service channels are used for *non-safety* service messages [3]. Beyond this, [2] proposes the following policy: "when the service channel is overloaded and the control channel is free, messages within the service queue are switched to the control one, and then considered as high priority messages."

c) Cooperative message transmission: In order to reduce the delay in sending high priority messages, [2] adopts 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 one with the highest priority within the channel is selected.

A bandwidth sharing strategy is defined for this congestion control method. Each node informs its neighbours about the priority of the messages it sends. If a node is notified about other messages from other nodes with a higher-priority, it delays the transmission of its own message. Also, it is assumed that half of the available bandwidth is reserved for the emergency messages to prevent any delay in transmitting such crucial information.

III. Probabilistic Model Checking

Formal verification have been extensively used to analyse various computational systems. In particular, *model checking* has been very useful in evaluating the correctness of complex systems, such as concurrent or distributed systems [5]. Model checking is an algorithmic technique which assesses whether a logical formula holds in a given model represented as finite structure. Actually, this logical formula is evaluated against all possible behaviours of the system, described by the finite model. In standard model checking formulae are expressed in a suitable *temporal* logic [6]. In order to analyse the *uncertain* and *unpredictable* behaviours of computer systems, *probabilistic* model checking was presented as a generalisation of the basic model-checking technique. Here, the system model is described in terms of either *Discrete-Time Markov Chains (DTMCs)*, *Continuous Time Markov Chains (CTMCs)* or *Markov Decision Processes (MDPs)*. Then, a formal specification of the system requirements is expressed in a probabilistic variant of temporal logic. One well-known logic is PCTL [7], which can express properties of the models containing probabilistic information. Its syntax includes standard classical operators such as $\wedge$ (and), $\vee$ (or) and $\Rightarrow$ (implies), and the *probabilistic operator* $\mathrm{P}_{\sim p}[.]$, where $p \in [0, 1]$ is a *probability bound* and $\sim$ is one of $<$, $\leq$, $\geq$, or $>$. Semantically:

> $\mathrm{P}_{\sim p}[\sigma]$ is satisfied in a state s if, and only if, the probability of taking a path from s satisfying the *path formula* σ in the interval is specified by the constraint '$\sim p$'.

PCTL also includes the temporal formulae $\mathrm{X}\phi$, $\Diamond\phi$, $\Box\phi$, $\phi\mathrm{U}\psi$, and $\phi_1\mathrm{U}^{\leq k}\phi_2$, with semantics:

> $\mathrm{X}\phi$ is true on a path if, and only if, ϕ is satisfied in the next state on that path;
> $\Diamond\phi$ is true on a path if, and only if, ϕ holds at some state on the path;
> $\Box\phi$ is true on a path if, and only if, ϕ holds at all states on the path;
> $\phi\mathrm{U}\psi$ is true on a path if, and only if, ϕ holds until ψ holds on the path; and
> $\phi_1\mathrm{U}^{\leq k}\phi_2$ is true if, and only if, on the path ϕ_2 satisfied within k time-steps and ψ_1 is true up until that point.

For example, $\mathrm{P}_{\geq x}[\varphi\ \mathrm{U}^{<\infty}\psi]$, states that "the probability of φ being true up until ψ occurs is greater than x"

Prism is a probabilistic model-checker that takes models of the forms outlined above (DTMCs, CTMCs, or MDPs) together with a property/requirement in PCTL. The Prism model-checking process then allows us to query the probability of a certain property being true. In addition to this, Prism also supports quantitative structures defining *costs* and *rewards*. These structures can be used to reason about quantitative measures such as "expected number of hits", "expected success rate", etc. This is achieved using 'R' operator, which works in a similar fashion to the 'P' operator above. For example, $\mathrm{R}^{=?}[\mathrm{C}^{\leq 10}]$ returns the expected cumulative reward within 10 units of operation.

IV. Formal Modelling

We now evaluate the efficiency and correctness of the earlier congestion protocol by applying the probabilistic model checking approach described above. The key characteristics of the congestion protocol were outlined in Section II. For the system model we also assume the following:

1) The number of vehicles within an interference range at any time can be any value in $\{0, .., n\}$, where n denotes the maximum number of vehicles. A probabilistic state machine modelling this is given in Fig. 1, illustrating how the "number of vehicles" is defined as a random number from $\{0, .., n\}$ with a uniform probability.
2) We differentiate three types of messages, which can be sent to/received from other nodes: *(i)* emergency messages driven by an event; *(ii)* periodic safety messages; and *(iii)* periodic non-safety service messages. Occurrence of an emergency message is unpredictable because it is event driven. The probabilistic state machine assigning the "emergency generation rate" to be

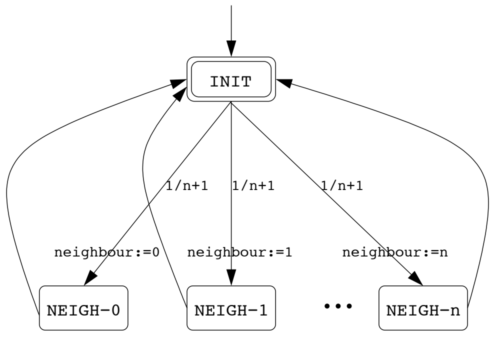


Fig. 1: Allocation of the number of neighbour nodes.

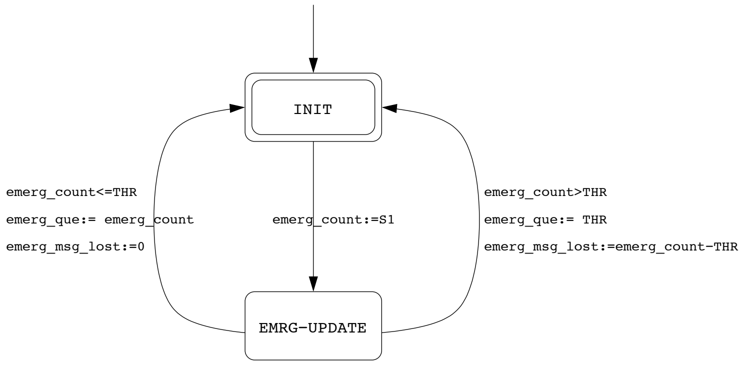


Fig. 2: Transition system for emergency message control.

from $\{0,..,m\}$ is similar to Fig. 1. On the other hand, we don't generate a probabilistic machine for safety and non-safety messages; we instead assume that each vehicle broadcasts constant safety and non-safety messages (Maximum values assumed are k and l, resp.).

3) We also assume two shared media. The control channel (CCH) is used for the transmission of emergency and safety messages, and the service channel (SCH) is used for the transmission of the service messages. 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.

Based on the characteristics of the congestion protocol and the assumptions above, we can now model the protocol. One approach to modelling is to instantiate a transition system for each message and then take the product of these to consider the behaviour of the overall system. However, this approach can be very expensive as the size of the combined transition system becomes large, making verification experiments very difficult. Instead, we adopt the *counting abstraction* or *population modelling* approach where we model just one transition system but we add a counter for each message and channel type. This abstraction approach is very suitable for our scenario since there are many identical and independent processes, whose overall behaviour can be captured by this approach.

Fig. 2 illustrates a state machine for emergency messages. The state machine basically updates the emergency message queue, and counts the number of emergency messages lost. In the figure, $\mathtt{S1} \doteq \mathtt{emerg\_que} + \mathtt{emerg\_msg} - min(\mathtt{emerg\_que}, \mathtt{MSG\_TRNS}/2)$ denotes the new queue size. 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 queue has less than `MSG_TRNS`/2 messages, it is cleared).

If the queue size is less than the threshold ($\mathtt{emerg\_count} \leq \mathtt{THR}$), then no message is lost ($\mathtt{emerg\_msg\_lost} := 0$); otherwise excess messages are lost ($\mathtt{emerg\_msg\_lost} := \mathtt{emerg\_count} - \mathtt{THR}$). Here we denote the message transmission rate by `MSG_TRNS`/2 msg/s, as we use half the control channel for emergency messages.

Transition systems for safety and service messages are similar to the one in Fig. 2; but we consider that if the service channel is overloaded and no safety messages are waiting in the queue, then any excess service messages are forwarded to the control channel. If the service channel is overloaded and some safety messages are waiting in the queue then the excess service messages are lost.

Channel bandwidth	10 MHz
Data rate	6 Mbps
Message transmission rate (`MSG_TRNS`)	1500 msg/s
Channel capacity (`CHNL_SIZE`)	1500 msg
Queue threshold (`THR`)	4500 msg
Average message size	500 bytes
n	50
m	100
k	100
l	200

TABLE I: Configuration parameters.

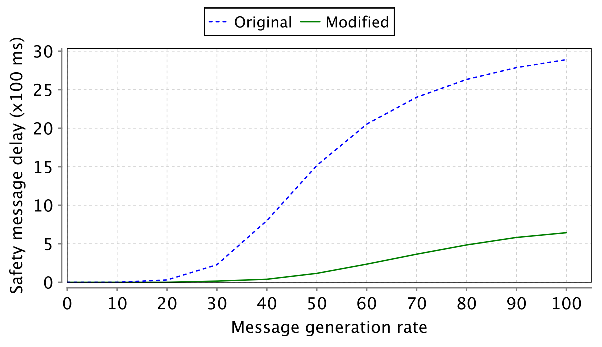
V. Verification and Refinement

We modelled our interpretation of the protocol from [2] and so generated transition systems in PRISM according to the configuration parameters given in Table I. PRISM takes the cross product of these transition systems to construct the overall state space. If we had modelled individual copies of each vehicle's and message's behaviour, and then taken the product, then the size of the resulting model would be *huge*. Using the counting abstraction approach significantly reduces (indeed, by *several orders of magnitude*) the state space required, and so makes the verification tractable.

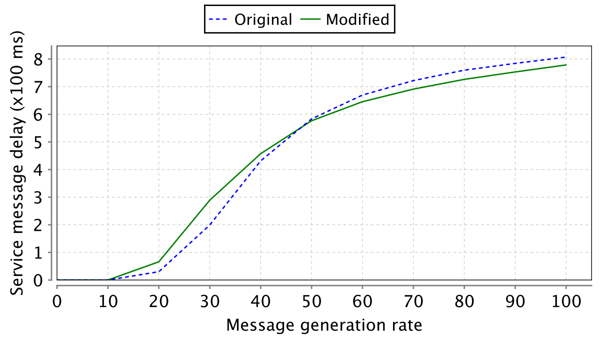
Based on this model we can verify the correctness of the protocol and evaluate the performance with respect to selected PCTL properties. We checked several properties, such as

(I) "emergency messages are never lost"
$\mathrm{P}^{=0}\Diamond(s = Lost_{emerg\_msg})$ and
(II) "emergency messages are never delayed"
$\mathrm{P}^{\geq 1}\Box(\mathsf{delay}_{emerg\_msg} = 0)$.

PRISM returned TRUE for both properties, showing that the congestion control protocol (with the above assumptions)

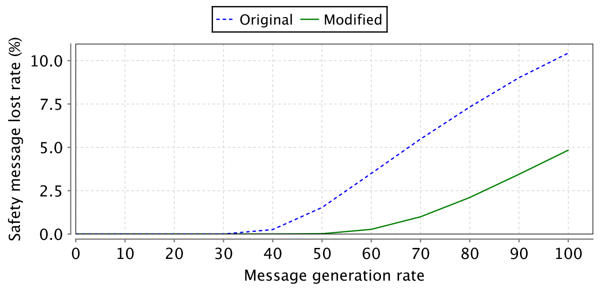


(a)

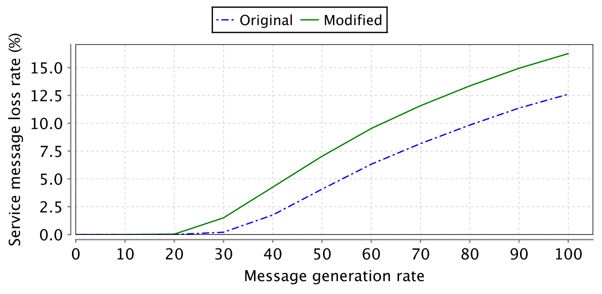


(b)

Fig. 3: Message delay *versus* message generation rate.



(a)



(b)

Fig. 4: Message loss rate *versus* message generation rate.

correctly works for emergency messages. However, the verification results showed us that the protocol does *not* guarantee some important properties. For example, the property $\mathrm{P}^{\geq 1}\Box(\mathsf{delay}_{sfty\_msg} < \mathsf{delay}_{srv\_msg})$ meaning

(III) "the delay of a safety message is always less than the delay of a service message"

when checked against the model using PRISM, returned FALSE indicating that, in at least one possible execution, this is *not* true. From Fig. 3 we see that the delays observed in safety messages are much higher than the delays for service messages (the verification results of the original congestion protocol is shown by the dashed lines). Similarly, we checked the loss rate for safety messages and service messages. As Fig. 4 shows, although the loss rate of safety messages is lower than the loss rate of service messages, the loss rate for safety messages remains high, causing the some critical messages to be lost. Our analysis also showed that the control channel is not effectively used since the bandwidth devoted to emergency messages is wasted most of the time.

Through such observations, obtained using the verification method, we then propose the following modifications: *(a)* periodic safety messages use all available bandwidth within the control channel; *(b)* when an emergency message arrives, it takes precedence and is transmitted without any delay; *(c)* service messages only use the service channel — if this channel is overloaded, the control channel is *not* used for service messages; and *(d)* the available bandwidth for both control and service messages is equally shared between nodes within an interference range. These modifications allowe us to reduce the model size even further.

We modelled the resulting state machines in PRISM, and performed verification. The results confirmed that the properties I, II and III above are all TRUE for the refined model. Fig. 3 and Fig. 4 show that, although the modifications we propose are not major, both the message delay and message loss rate are improved (The verification results of the modified congestion protocol is shown by the solid lines). As seen in Fig. 4(b), the only exception to this is the "service message loss rate", which is slightly increased over that of the original method. Since service messages do not carry critical information, this increase does not affect the network safety.

In the scenario above we assumed that both safety and non-safety (service) messages are sent periodically, and we therefore considered that they stay constant in each execution. Although this is almost standard in vehicular technology, the average number of messages transmitted will be different for each vehicle. We therefore refine the scenario and assume that the average number of safety and non-safety message generation rates per vehicle per time instant are in $\{0, .., K\}$ and $\{0, .., L\}$, respectively. Namely, the average numbers are randomly assigned to any value within the respective ranges. We model this as a transition system similar to Figure 1.

Using the resulting probabilistic model we can also query some properties via PRISM, and produce results without plotting any explicit graph. Specifically, the properties together with their PCTL expressions are given in Table II. Note that the question mark, '=?', over the operators P and R asks PRISM to return a numeric value for the property considered. For example, $\mathrm{P}^{=?}$ asks the *actual* probability that some behaviour of a model is observed. In Table II, $\mathsf{delay}_{sfty\_msg}$, delay_{srv_msg}, ctrl_que, srv_que, and s, are state variables. In properties 1–6, the corresponding state variables

Property	Informal and Formal Specification	PRISM verif.
1	Probability of a safety message being lost: $\mathrm{P}^{=?}\Diamond(\mathsf{s} = Lost_{sfty\_msg})$	0.79
2	Probability of a service message being lost: $\mathrm{P}^{=?}\Diamond(\mathsf{s} = Lost_{srv\_msg})$	0.89
3	Probability of a safety message being delayed for 0.1 seconds: $\mathrm{P}^{=?}\Box(\mathsf{delay}_{sfty\_msg} \leq 0.1)$	0.21
4	Probability of a service message being delayed for 0.1 seconds: $\mathrm{P}^{=?}\Box(\mathsf{delay}_{srv\_msg} \leq 0.1)$	0.10
5	Probability of never being a queue in the control channel: $\mathrm{P}^{=?}\Diamond\ \forall\Box(\mathsf{ctrl\_que} \leq \mathtt{THR})$	0.19
6	Probability of never being a queue in the service channel: $\mathrm{P}^{=?}\Diamond\ \forall\Box(\mathsf{srv\_que} \leq \mathtt{THR})$	0.09
7	Expected loss rate of a safety message in steady-state: $\mathrm{R}^{=?}_{sfty\_msg\_loss}[\mathbf{S}]$	1.63%
8	Expected loss rate of a service message in steady-state: $\mathrm{R}^{=?}_{srv\_msg\_loss}[\mathbf{S}]$	8.48%
9	Expected delay of a safety message in steady-state: $\mathrm{R}^{=?}_{sfty\_msg\_delay}[\mathbf{S}]$	0.29 seconds
10	Expected delay of a service message in steady-state: $\mathrm{R}^{=?}_{srv\_msg\_delay}[\mathbf{S}]$	0.52 seconds
11	Loss rate of a safety message does not exceed convergence value: $\bigwedge_{i=1}^{i=8}(\mathrm{R}^{=?}_{sfty\_msg\_loss}\mathbf{C}^{\leq 10^i} \leq 1.63)$	TRUE
12	A safety message is not lost within an hour': $\forall[(s \neq Lost_{sfty\_msg})\mathrm{U}^{\leq 3600}(s = Lost_{sfty\_msg})]$	FALSE

TABLE II: Informal and formal specification of properties and PRISM verification results.

are compared against a value at the paths and states specified by the operators of the corresponding formula. In properties 7–10, '$\mathbf{S}$' is an operator which denotes the *steady state* (*long-run* or *equilibrium*) behaviour of a model [12]. Meanwhile, properties 11–12 denote sample Boolean properties.

VI. Conclusion and Future Work

This paper formally analyses (our understanding of) the congestion control approach for vehicular ad hoc networks introduced in [2]. (Since the paper does not define all details, there remains the possibility of errors in interpretation.) Using probabilistic model checking we evaluate the correctness and efficiency of the proposed protocol. In order to reduce the state space and so to retain a tractable problem we use a "counting abstraction" approach in modelling the system. As seen in the example specifications in Table II, we can analyse a probabilistic model through PCTL and similar logics, providing a wide range of properties to analyse the correctness and effectiveness of a system. The results show that this technique is very efficient even if we assume high numbers of messages and neighbour nodes. From the verification results we observed that by making some simple changes to the protocol we can improve both channel usage and message transmission. Specifically, we proved that the modifications we propose allow us achieve a better loss rate and delay for safety messages. This study therefore shows how verification techniques can be used to evaluate and enhance the efficiency and correctness of a VANET protocol.

The work in this paper can be extended in several dimensions. In this paper, we assumed three priority levels for messages. One dimension is to extend these priority levels. It will be also interesting to increase the number of service channels considered. We are also planning to formally analyse topology changes within VANET environments.

Finally, we acknowledge support from the EPSRC via research project EP/F033567.

References

[1] J. Bengtsson, K. Larsen, F. Larsson, P. Pettersson, and W. Yi, "UPPAAL—A Tool Suite for Automatic Verification of Real-time Systems," in *Proc. DIMACS/SYCON Workshop on Hybrid Systems III: Verification and Control*, 1996, pp. 232–243.

[2] M. Bouassida and M. Shawky, "A Cooperative and Fully-distributed Congestion Control Approach within VANETs," in *Proc. 9th ITST*, 2009, pp. 526–531.

[3] C. Campolo, A. Cortese, and A. Molinaro, "CRaSCH: A Cooperative Scheme for Service Channel Reservation in 802.11p/WAVE Vehicular Ad hoc Networks," in *Proc. ICUMT Workshops*, 2009, pp. 1–8.

[4] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella, "NuSMV 2: An OpenSource Tool for Symbolic Model Checking," in *Proc. Conf. Computer Aided Verification*. Springer, 2002, pp. 241–268.

[5] E. Clarke, O. Grumberg, and D. Peled, *Model Checking*. MIT Press, 1999.

[6] E. A. Emerson, "Temporal and Modal Logic," in *Handbook of Theoretical Computer Science*. Elsevier, 1990, pp. 996–1072.

[7] H. Hansson and B. Jonsson, "A Logic for Reasoning about Time and Reliability," *Formal Aspects of Computing*, vol. 6, pp. 102–111, 1994.

[8] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker, "PRISM: A Tool for Automatic Verification of Probabilistic Systems," in *Proc. 12th TACAS*. Springer, 2006, pp. 441–444.

[9] G. J. Holzmann, *Design and Validation of Computer Protocols*. Prentice-Hall, 1991.

[10] "IEEE P802.11p/D3.0, Draft Amendment for Wireless Access in Vehicular Environments (WAVE)," 2007.

[11] C. Jin, D. Wei, and S. Low, "FAST TCP: Motivation, Architecture, Algorithms, Performance," in *Proc. 23rd INFOCOM*, vol. 4, 2004, pp. 2490–2501.

[12] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM: Probabilistic Symbolic Model Checker," in *Comp. Perf. Evaluation: Modelling Techn. and Tools*, ser. LNCS. Springer, 2002, vol. 2324, pp. 113–140.

[13] A. Lomuscio, B. Strulo, N. G. Walker, and P. Wu, "Model Checking Optimisation Based Congestion Control Models ," *Fundamenta Informaticae*, vol. 102, no. 1, pp. 77–96, 2010.

[14] M. Torrent-Moreno, P. Santi, and H. Hartenstein, "Fair Sharing of Bandwidth in VANETs," in *Proc. 2nd ACM International Workshop on Vehicular Ad hoc Networks (VANET)*. ACM Press, 2005, pp. 49–58.

[15] L. Wischhof and H. Rohling, "Congestion Control in Vehicular Ad hoc Networks," in *Proc. IEEE International Conference on Vehicular Electronics and Safety*, 2005, pp. 58–63.

[16] S. Yousefi, M. S. Mousavi, and M. Fathy, "Vehicular Ad Hoc Networks (VANETs): Challenges and Perspectives," in *Proc. 6th International Conference on ITS Telecommunications*, 2006, pp. 761–766.