Scalable VANET Simulations with NS-3

Ricardo Fernandes and Michel Ferreira
Instituto de Telecomunicações, DCC/FC - University of Porto
Rua Campo Alegre, 1021/1055, 4169-007 Porto, Portugal
Email: {rjf, michel}@dcc.fc.up.pt

***Abstract*—The study of large scale scenarios in vehicular ad-hoc networks is of critical importance to the design of communication protocols and applications. Since management and operation of a large number of real nodes is unpractical, simulation is considered as the most viable methodology to evaluate such large scale scenarios. However, the widely used simulator NS-3 has limited performance and scalability, causing the simulation of medium scale networks with more than 1,000 nodes to become very hard. In this paper we identify the sources of such performance limitations and propose possible improvements to the physical and mobility layers of NS-3. Our results show that the proposed NS-3 optimization leads to significant performance improvements both in execution time and memory usage, making it feasible to run simulations with more than 10 000 nodes.**

I. Introduction

Vehicular communications promise a myriad of new applications for road safety, traffic efficiency and infotainment. Due to the inherently self-organizing and mobility features of vehicular ad-hoc networks, there is a growing research interest in large-scale scenarios where thousands of vehicles access the network. Consequently, there is a need for the performance evaluation of new protocols and distributed applications in such large-scale environments. For instance, applications addressing traffic control in metropolitan scenarios [1] and the study of the inherent network protocols are of great interest today. However, setting-up a real vehicular network with this degree of scalability is an expensive and unpractical task. Thus, simulation is considered the most viable methodology to design and evaluate large-scale vehicular network applications and protocols.

NS-3 [2] is discrete-event network simulator widely used and well accepted by the research community. It provides various protocol modules and has a stable design for improved scalability, being capable of simulating a large number of nodes and high traffic densities. The study in reference [3] demonstrated that NS-3 delivers the best overall performance of several network simulators. However, for the simulation of large-scale wireless networks, the performance is limited to a few hundred nodes. In [4], we presented a simulation framework with a very tight integration between NS-3 and the microscopic traffic simulator DIVERT [5]. Although it is a valuable tool for the research community, our results in that previous work have shown that the overall performance and scalability of such a coupled framework is severely limited by the performance and scalability of NS-3. Thus, the main goal of the work herein presented is to extend our previous framework, identifying the sources of performance limitation and proposing possible solutions.

When simulating a vehicular ad-hoc network, nodes generally move in roads and communicate with neighbors which are located within their communication range. In the case of a wireless transmission, the simulator has to determine which nodes will receive the transmitted packet. Scalable simulations require efficient methods for determining which nodes are reachable by other nodes and to predict when a node will come within, or go beyond, the communication range of another node. In the case of NS-3, for a simulation of N nodes sharing the same channel, all the N nodes are stored in a list. To locate the nodes within the communication range of a given transmitter node, the entire list must be traversed, originating $O(N)$ iterations. As transmissions are usually performed repeatedly, this process could dominate the computation in the simulation and effectively limit the scalability when a large N is considered.

In this work we evaluate possible enhancements to the physical layer and mobility models of NS-3. We propose a spatial indexing data structure to efficiently store and update nodes, as well as to allow fast neighbor finding within a given range from a source node.

The remainder of this paper is organized as follows. The following Section discusses the related work in network simulation and its performance. In the Section III, we describe the wireless module of NS-3 and identify the causes that limit its performance. In Section IV, we describe our improvements to the wireless module. The evaluation of the proposed improvements is described in Section V, along with the performance results and discussion. Finally, in Section VI the main conclusions are given.

II. Related Work

Discrete-event simulation is the most used technique to evaluate protocols and architectures for wireless networks. The existing network simulators, differ not only in realism and accuracy of the results, but also in performance and scalability of simulations. Naturally, as the simulation complexity increases in an attempt to closely mimic reality, the simulation performance decreases. Performance comparison between network simulators was given in [6] and [3]. However, results demonstrated that current simulators have performance limitations when dealing with large-scale scenarios.

A common problem in wireless network simulations is to determine which nodes are involved in the reception of

978-1-4673-0990-5/12/$31.00 ©2012 Crown

a given transmitted packet. Since a wireless signal strength decreases with distance [7], for a large scenario, only a small subset of nodes will be within reception range and the remaining network will not be affected by the transmission. Perrone and Nicol show in [8] that techniques devised for the simulation of systems of self-gravitating bodies (N-body problem) can be successfully applied to reduce the complexity of interference computations in wireless network simulations. Compared to the brute-force approach, results show that the Barnes-Hut [9] algorithm significantly reduces the number of computed pairwise interactions between nodes. Exploiting the idea of limited interference, [10] proposed two ways to enhance NS-2 performance, both based on additional data structures to keep track of the nodes - a Grid-based and a List-based node organization. The improvements resulted in a substantial increase of performance, allowing simulations of up to 3,000 nodes and 30 times faster. The authors of [11] also propose spatial data structures for efficient radio signal propagation, and [12] describes another grid-based approach where the computation is staged to allow the reuse of previously computed results. In [13], authors also propose tailored data structures and new techniques to enhance the proximity detection of mobile nodes. Performance improvements through the use of distributed and parallel computing architectures are also gaining a lot of interest in the simulation of large-scale wireless networks [14], [15], [16].

III. Problem Statement

NS-3 is a discrete-event network simulator widely used by the research community, succeeding the popular ns-2 but with a simulation architecture redesigned from scratch. The goal was to create a new network simulator aligned with modern research needs and to develop it in an open-source community. It provides a number of realistic protocol modules and a stable design.

After a deep analysis on the performance of the NS-3 wireless module, we noticed that the execution of the channel propagation module strongly dominates the computation of simulations. The NS-3 implementation of 802.11 is inherited from Yans [17] and many features and design issues proposed in Yans determined the current NS-3 architecture. As illustrated in Fig. 1, the signal propagation is computed within the *WifiChannel* (medium), which contains a list of all the *WifiPhy* (radios) that share the same channel. The *WifiPhy* implements the 802.11 PHY layer model, allowing signal transmissions and receptions through the channel.

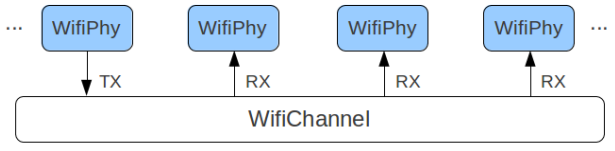

Fig. 1: List-based data structure used by the NS-3 wireless channel. The *WifiPhy* nodes are the elements of the list.

Consider a *WifiPhy* willing to do a transmission on the shared medium. Packets are pushed to *WifiChannel* together with a transmission power P_t, and for each receiver, the signal delay D and power loss P_{loss} are calculated by a propagation delay model and a propagation loss model, respectively. Then, a new event with delay D is scheduled for packet reception on the receiver. Only upon the execution of this event, is checked whether the transmission occurs or not. If the received power P_r is below the receiver's Energy Level Threshold (EDT), packets are dropped. Thus, for each transmission, the simulator schedules a reception event on every *WifiPhy* in the *WifiChannel*. This causes a large number of unnecessary steps, which increase the cost of the computation exponentially in N, since a network with N pairs of transmitters and receivers requires $O(N^2)$ pairwise interactions to be computed. Moreover, due to this number of scheduled events, the inherent memory allocations severely limit the scalability of simulations.

P_r is given by the Equation 1, where G_t and G_r are the antenna gains for the transmitter and the receiver respectively. P_{loss} reflects the power loss (negative number) between the two nodes and is given by one of the possible propagation loss models of NS-3. Gain is given in units of dB, and power is given in units of dBm.

$$P_r = P_t + G_t + G_r + P_{loss} \tag{1}$$

IV. Improvements on the Network Simulator

Modeling propagation efficiently is essential to improve the scalability of wireless simulations. During a signal transmission, the *WifiChannel* must deliver the signal to all the affected *WifiPhys* in an efficient manner. Since wireless signal strength decreases with distance, there is a limit range R to a receiver node, such that a power signal originated from nodes outside R is below the EDT. Thus, for a large scenario, only a small subset of nodes within R will be affected by the transmission and the remaining nodes should be ignored, as they are not affected by the transmission.

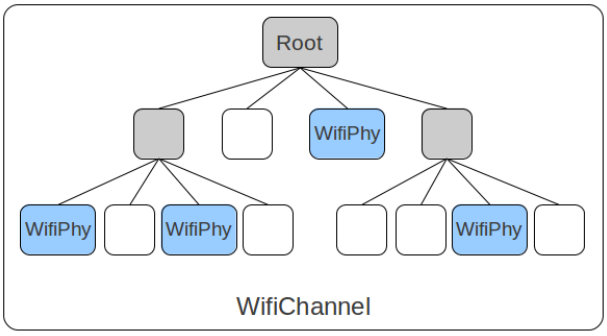

Fig. 2: Quadtree based Channel

Fig. 3: Quadtree-based data structure of our proposed wireless channel. The *WifiPhy* nodes are stored in the leaves of the tree.

For a given *WifiPhy* that intents to transmit a signal, the *WifiChannel* must know which nodes are within R and select

them as signal receivers. This process should be efficient in terms of computational complexity. Moreover, due to the nature of vehicular networks, nodes are constantly moving, which means that for a given transmitter, the nodes within R change in time.

In order to allow fast node search within a limited range, we propose the use of a spatial indexing data structure [18] to store the nodes in the network. Since nodes in a mobile wireless network can be seen as points moving in a scenario, our modified *WifiChannel* uses a Quadtree-based data structure [19] to store *WifiPhys*. Thus, each *WifiPhy* has a position associated, which is already implicit since it is attached to an NS-3 node. Note that for 3-dimensional coordinates we should use an Octree [18], but in this work we are only interested in 2-dimensional scenarios.

Our Quadtree is a spatial index that divides the scenario bounding box into homogeneous cells of regularly decreasing size. Each decrement in size is $1/4$ of the area of the previous cell. The tree segmentation process continues until a maximum level of depth is reached and the *WifiPhy* nodes are stored in the leaves as illustrated in Fig. 3. Each *WifiPhy* knows its location in the tree, i.e, knows the exact leaf where it is contained. Moreover, each node in the tree has a pointer to its parent node.

Whenever a new *WifiPhy* is added to the *WifiChannel* it is inserted in the tree. For insertion we use the traditional insertion algorithm for Quadtrees [18]. The *WifiPhy* descends from the root and traverses the tree selecting the cells that contain its location, until the correspondent leaf is reached.

Although a Quadtrees are designed for static points, the mobility of the nodes results in constant changes in the structure of the tree. Thus, the tree must always be updated in order to provide fast and accurate neighbor finding.

A. Updating the Tree

Since a propagation loss model depends on the distance between the nodes involved in the communication, knowing the exact node's location is crucial for the accuracy of the signal transmission. However, in a vehicular network vehicles are permanently moving in the scenario and consequently the topology of the network changes over time. Thus, the structure of the tree suffers constant changes during the simulation.

Based on the mobility features of a vehicular network, we propose a fast method for tree updating in response to a vehicular movement. Typically, each vehicle with a realistic mobility model only moves a very small distance at each simulation step. For instance, a vehicle traveling at a speed of $100km/h$, for each second of simulation, travels a distance of 27 meters. This shows that the probability of a *WifiPhy* move from the current node to one of the closest nodes is high. Thus, whenever a *WifiPhy* receives a notification of a position update, it tests if the bounding box of the current node still contains the new position. If so, the tree does not need to be updated, otherwise the *WifiPhy* is removed from the current leaf and ascends to the parent node to test if the bounding box of any other child contains the new position.

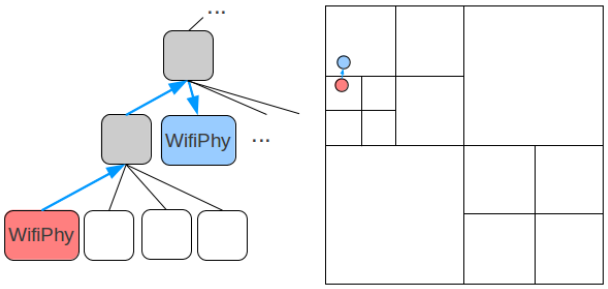

Fig. 4: The operation of moving a node in the tree in a response to a vehicular movement. The red node indicates the previous location and the blue node the location after the movement.

The *WifiPhy* ascends until it finds a valid node, and then it is inserted in the tree, descending from the current node to the correspondent leaf. Due to the intrinsic features of vehicular movements, usually each node only needs to ascend one or two levels at most, reducing the overhead of updating the tree. This process is illustrated in Figure 4.

B. Neighbor Finding

Whenever a *WifiPhy* intents to transmit a signal, the *WifiChannel* should efficiently search for the other *WifiPhys* within R and select them as potential receivers. Usually, range queries in a Quadtree would start from the root node. The algorithm descends the tree to the leaves, visiting only the nodes that intersect the communication range R.

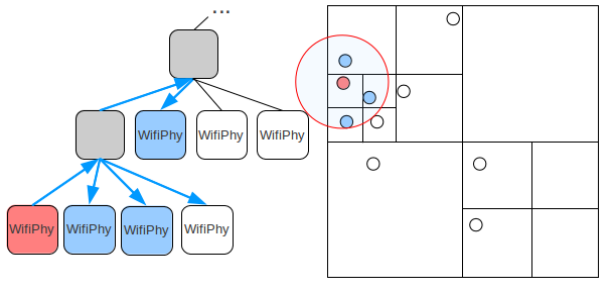

Fig. 5: The operation of filtering the neighbors within the communication range R, during a signal transmission originated by a *WifiPhy*. The red node is the sender and the blue nodes are the receivers.

Because of the inherently spatial division of the tree, for each *WifiPhy* in a leaf, the closest neighbors are within the closest nodes of the tree. Thus, whenever a *WifiPhy* intents to do a transmission, we start from its leaf and climb until we find a node with a bounding box that contains the entire communication range R. During the climb, only the nodes with a bounding box that intersects R are visited. As we reach the leaves, only the *WifiPhys* within R are selected as potential receivers. We say potential receivers, because the received power signal will be further evaluated by the physical model. This process is illustrated in Fig. 5.

By filtering the *WifiPhys* within R, we avoid checking additional receivers for which we know the reception will fail. However, the R value must be chosen carefully to avoid false negatives. In Equation 1, in addiction to the P_{loss} value, P_r not only depends on P_t and G_t, but also on the G_r. Thus, the communication range R not only depends on the transmission power and antenna gains of the senders, but also on the antenna gains of the receivers, and in a simulation scenario we may find receivers with different antenna gains.

For a signal transmission from a transmitter A to a receiver B, the transmission only succeeds at B if $P_r > EDT$. Thus, there is a maximum value of accepted P_{loss} for which B accepts the transmission:

$$P_{loss} = EDT - (P_t + G_t + G_r) \tag{2}$$

Depending on the propagation loss model, we can then calculate the maximum distance R from the maximum accepted P_{loss}. For instance, for the Friis propagation model [20] which gives the power received under ideal conditions, the P_{loss} is given by:

$$P_{loss} = 10 \log_{10} \left[\frac{\lambda^2}{(4\pi R)^2 L} \right] \tag{3}$$

Then, the maximum distance R is given by:

$$R = 10^{\frac{1}{20}\left[10 \log_{10}(\lambda^2) - 10 \log_{10}(4^2 \pi^2 L) - P_{loss}\right]} \tag{4}$$

Knowing the maximum possible values for EDT and G_r between all the receivers in the *WifiChannel*, by Equations 2 and 4 we can calculate the maximum range R which contains all the potential receivers for a given transmitted signal.

V. Evaluation

The goal of this section is to compare the performance between our improved simulator and the original version of NS-3. For a precise evaluation that allows us to analyze the limits in terms of performance, we require a typical network protocol and a large-scale scenario with a realistic mobility model. Thus, we fed the simulator with the real urban map of the city of Porto, which has an area of $41.3km^2$, and implemented the beaconing mechanisms commonly used by VANET applications. In our scenario, each vehicle arbitrarily selects one of the predefined routes based on a shortest-path algorithm between an origin and a destination. During the trip, vehicles periodically broadcast beacons to the neighborhood warning their presence. The idea is to provide location-based information to the drivers, allowing them to know the location of all the vehicles in the surrounding environment.

Experiments were carried out by the framework described in [4] using the version 3.12.1 of NS-3. We used a PC with a 2.6 GHz Intel Core2 Duo CPU, 4Gb RAM and the Ubuntu 11.04 operating system. The interval for beaconing was 1 second and the size of each packet was 50 bytes. We performed simulations with a variable number of nodes in ranges from 500 to 15,000 vehicles. Please note that at a given time of the day, the city of Porto has approximately 10,500 vehicles [21]. For each simulation, we measured the processing time and memory usage of the channel propagation method of NS-3, and also of our channel propagation method for 100, 250, 500 and 1,000 meters as the maximum range R.

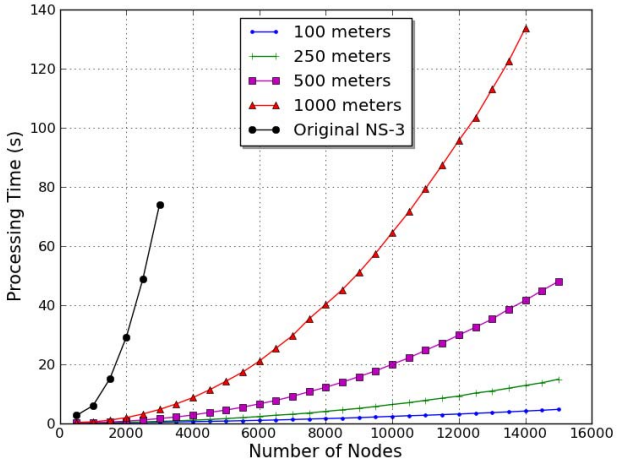

Fig. 6: Processing time required to execute a simulation step of 1 second.

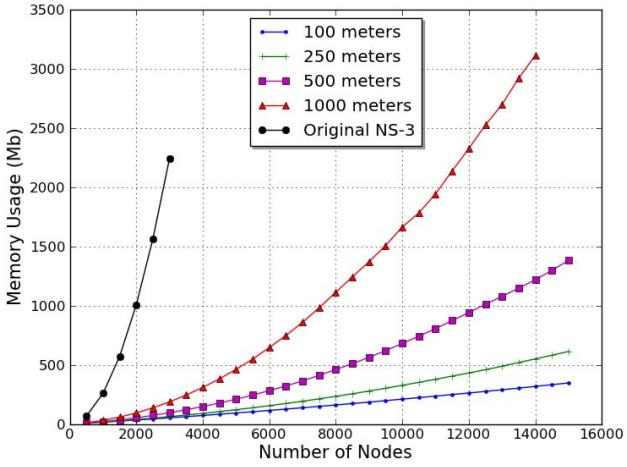

Fig. 7: Memory usage.

Figure 6 shows the results for the metric "processing time required to execute a simulation step of 1 second" and Fig. 7 shows the results for the memory usage. We can see that using the original NS-3 the processing time and memory usage grow exponentially with the number of nodes. This was expected due to the $O(N)$ interactions and scheduled events for every transmitted packet. In fact, due to memory restrictions, it was not possible to evaluate the original NS-3 for more than 3,000 nodes. However, using our proposed channel propagation scheme, the gains in terms of efficiency are very significant. As depicted in Fig. 6 and 7, the processing

time is much faster and the memory consumption is much lower, enabling simulations with higher number of nodes. For instance, it is interesting to note that for $R = 250$, the processing time required to execute a step with 3,000 vehicles, is less than 1 second. This means that, in a scenario of 3,000 nodes where the maximum communication range is 250 meters, the simulation would run in real-time. However, if we wanted to perform the same simulation with the original NS-3, each step of simulation would take 74 seconds.

The results also demonstrate that for small values of R the processing time becomes faster and the memory consumption becomes lower. On the other hand, as R increases, the processing time becomes slower and the memory usage becomes higher. At the worst case, when R is so large that all the nodes can reach any other node in the network, the performance is comparable to the performance of the original NS-3, with a small degradation due to the overhead introduced by our algorithms. Fig. 8 shows a snapshot of the internal structure of the Quadtree during a simulation.

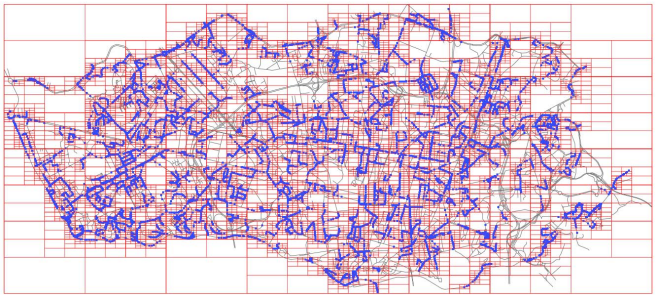

Fig. 8: Snapshot of the Quadtree during a simulation. The red lines represent the quadrants and the blue points represent the *WifiPhys* in the leaves. The roads are in background.

VI. Conclusions

NS-3 simulator has limited performance for the simulation of large-scale wireless networks, such as vehicular ad-hoc networks. In order to increase the performance and scalability, we proposed improvements to the wireless module of NS-3, through the use of a spatial indexing data-structure and algorithms for fast position updates and neighbor finding. Our results show the optimization herein proposed makes it feasible to run simulations with more than 10,000 nodes, resulting in significant performance improvements both in execution time and memory usage. At the worst case, when all the nodes can communicate with any other node in the network, the performance is comparable to the performance of the original simulator.

Acknowledgment

The authors would like to thank the financial support provided by the Portuguese Foundation for Science and Technology (FCT) under the doctoral grant (SFRH/BD/61676/2009) and also the projects DRIVE-IN (CMU-PT/NGN/0052/2008) and VTL (PTDC/EIA-CCO/118114/2010)

References

[1] M. Ferreira, R. Fernandes, H. Conceição, W. Viriyasitavat, and O. K. Tonguz, "Self-organized traffic control," in *Proceedings of the seventh ACM international workshop on VehiculAr InterNETworking*. ACM, 2010, pp. 85–90.

[2] T. R. Henderson, S. Roy, S. Floyd, and G. F. Riley, "NS-3 Project Goals," in *WNS2 '06: Proceeding from the 2006 workshop on ns-2: the IP network simulator*. ACM, 2006, p. 13.

[3] E. Weingartner, H. Vom Lehn, and K. Wehrle, "A performance comparison of recent network simulators," in *Communications, 2009. ICC'09. IEEE International Conference on*. IEEE, 2009, pp. 1–5.

[4] R. Fernandes, P. d'Orey, and M. Ferreira, "Divert for realistic simulation of heterogeneous vehicular networks," in *Mobile Adhoc and Sensor Systems (MASS), 2010 IEEE 7th International Conference on*. IEEE, 2010, pp. 721–726.

[5] H. Conceição, L. Damas, M. Ferreira, and J. Barros, "Large-scale simulation of V2V environments," in *Proceedings of the 2008 ACM symposium on Applied computing*. ACM, 2008, pp. 28–33.

[6] F. Kargl and E. Schoch, "Simulation of MANETs: a qualitative comparison between JiST/SWANS and ns-2," in *Proceedings of the 1st international workshop on System evaluation for mobile platforms*. ACM, 2007, pp. 41–46.

[7] J. Parsons and P. Parsons, *The mobile radio propagation channel*. Wiley Online Library, 2000, vol. 81.

[8] L. Perrone and D. Nicol, "Using n-body algorithms for interference computation in wireless cellular simulations," in *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2000. Proceedings. 8th International Symposium on*. IEEE, 2000, pp. 49–56.

[9] J. Barnes and P. Hut, "A hierarchical o(n log n) force-calculation algorithm," *nature*, vol. 324, p. 4, 1986.

[10] V. Naoumov and T. Gross, "Simulation of large ad hoc networks," in *Proceedings of the 6th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems*. ACM, 2003, pp. 50–57.

[11] R. Barr, "Swans-scalable wireless ad hoc network simulator," *User Guide*, 2004.

[12] K. Walsh and E. Sirer, "Staged simulation: A general technique for improving simulation scale and performance," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 14, no. 2, pp. 170–195, 2004.

[13] L. Bononi, M. Bracuto, G. D'Angelo, and L. Donatiello, "Proximity detection in distributed simulation of wireless mobile systems," in *Proceedings of the 9th ACM international symposium on Modeling analysis and simulation of wireless and mobile systems*. ACM, 2006, pp. 44–51.

[14] X. Zeng, R. Bagrodia, and M. Gerla, "Glomosim: a library for parallel simulation of large-scale wireless networks," *ACM SIGSIM Simulation Digest*, vol. 28, no. 1, pp. 154–161, 1998.

[15] L. Bononi, M. Di Felice, M. Bertini, and E. Croci, "Parallel and distributed simulation of wireless vehicular ad hoc networks," in *Proceedings of the 9th ACM international symposium on Modeling analysis and simulation of wireless and mobile systems*. ACM, 2006, pp. 28–35.

[16] S. Bhatt, R. Fujimoto, A. Ogielski, and K. Perumalla, "Parallel simulation techniques for large-scale networks," *Communications Magazine, IEEE*, vol. 36, no. 8, pp. 42–47, 1998.

[17] M. Lacage and T. Henderson, "Yet another network simulator," in *Proceeding from the 2006 workshop on ns-2: the IP network simulator*. ACM, 2006, pp. 12–es.

[18] H. Samet, "Spatial data structures," *Modern Database Systems, The Object Model, Interoperability and Beyond*, pp. 361–385, 1995.

[19] R. Finkel and J. Bentley, "Quad trees a data structure for retrieval on composite keys," *Acta informatica*, vol. 4, no. 1, pp. 1–9, 1974.

[20] H. Friis, "A note on a simple transmission formula," *proc. IRE*, vol. 34, no. 5, pp. 254–256, 1946.

[21] M. Ferreira, H. Conceição, R. Fernandes, and O. K. Tonguz, "Stereoscopic aerial photography: an alternative to model-based urban mobility approaches," in *Proceedings of the sixth ACM international workshop on VehiculAr InterNETworking*. ACM, 2009, pp. 53–62.