2020

An Algorithm based on VANET Technology to Count Vehicles Stopped at a Traffic Light

Manuel Contreras
Central University of Venezuela

Eric Gamess
Jacksonville State University, egamess@jsu.edu

Recommended Citation

Contreras, M., & Gamess, E. (2020). An Algorithm based on VANET Technology to Count Vehicles Stopped at a Traffic Light. International Journal of Intelligent Transportation Systems Research, 18(1), 122. https://doi.org/10.1007/s13177-019-00184-3

An Algorithm based on VANET Technology to Count Vehicles Stopped at a Traffic Light.

Manuel Contreras. School of Computer Science, Central University of Venezuela, Los Chaguaramos, Caracas, Venezuela. e-mail: mcontre@ula.ve.

Eric Gamess. MCIS Department, Jacksonville State University, Jacksonville, AL, USA. e-mail: egamess@jsu.edu.

Manuel Contreras is the corresponding author.

An Algorithm based on VANET Technology to Count Vehicles Stopped at a Traffic Light

Abstract Vehicular Ad hoc Networks (VANETs) have gained considerable attention in the past few years due to their promising applicability in relation to the Intelligent Transportation Systems (ITSs). This emerging new technology will provide timely information to develop adaptive traffic light control systems that will allow a significant optimization of the vehicular traffic flow. In this paper, we introduce a novel algorithm for counting vehicles stopped at a traffic light using VANET technology. The algorithm is based on the idea of the propagation of a count request message from the RSU (originating unit) toward the vehicles that are at the end of the waiting line, and the propagation of a response message (with the number of vehicles counted) in the opposite direction, that is, from the vehicles at the end of the line toward the RSU. For this, our algorithm uses BEACON messages periodically to exchange the necessary information between any two 1-hop neighbors. Using the data received from BEACON messages, each vehicle can maintain its own neighbors list. To validate and evaluate the performance of our proposal, we use Veins (Vehicle in Network Simulation) and TraCI (Traffic Control Interface). The former is a framework that ties together a network simulator (OMNeT++) with a road traffic simulator (SUMO), and the latter is an API for the communications between both simulators by providing TCP connections between each other. The results of the simulations performed in different scenarios are encouraging since they indicate that the proposed algorithm efficiently computes a number of vehicles very close to the real one, using a few control messages.

Keywords: VANETs; Vehicular Networks; Vehicle Counting; OMNeT++; SUMO; Veins.

1 Introduction

Vehicular Ad hoc Networks (VANETs) are aimed at communications between vehicles [1]. They are similar to Mobile Ad hoc Networks (MANETs), where mobile units are vehicles, but with a very dynamic topology and density, high speed, and a mobility bounded by the road infrastructure and neighboring vehicles [2].

According to [3], the aim of VANETs is the development of platforms for communications between moving vehicles and between them and the road infrastructure. In VANET, there are two types of networking units: (1) On-Board Units (OBUs) are placed inside vehicles for communications to make them "smart objects" rather than mere transportation tools and (2) Road Side Units (RSUs) that are fixed and installed near the road. A VANET allows two types of communications: (1) communications between vehicles often referred to as Vehicle-to-Vehicle (V2V) communications that take place between OBUs, and (2) communications between vehicles and RSUs, known as Vehicle-to-Infrastructure (V2I). Both modes of communications can be performed using the same wireless communication technology, such as IEEE 802.11p [4]. Also, a VANET-enabled vehicle should be able to receive and relay messages to other VANET-enabled vehicles in its neighborhood (also known as multi-hop relaying) [1].

VANETs used short-range wireless communications (e.g., IEEE 802.11p [4]). A band of frequencies has already been reserved by the Federal Communications Commission (5.850 to 5.925 GHz), and it is generally divided into channels of 10 MHz: 6 Service CHannels (SCHs) and 1 Control CHannel (CCH) [4][5]. The SCHs are general purpose channels, that is, they can be used for safety applications or not. The CCH is reserved for safety applications. IEEE 802.11p also permits the aggregation of two contiguous channels, to form a wider channel with a bandwidth of 20 MHz. In the specialized literature, this band of frequencies designated by the Federal Communications Commission (FCC) is also known as Dedicated Short Range Communication (DSRC).

Wireless Access in Vehicular Environments (WAVE) has been proposed by the IEEE to specify the architecture for VANETs. It is based on several documents for standardization. The Physical (PHY) and the Medium Access Control (MAC) layers of WAVE are presented in the IEEE 802.11p standard. In the network layer, WAVE promotes the usage of two protocols: (1) Internet Protocol version 6 (IPv6) and (2) WAVE Short Message Protocol (WSMP). As known, IPv6 is the successor of IPv4, and it can be used for most of the applications. Unlike IPv6, WAVE is a very fast and light protocol, focused on supporting safety applications. To manage the seven channels that are not overlapped, IEEE 1609.4 [6] introduces the multi-channel operations.

VANETs support a large number of Intelligent Transportation System (ITS) applications, that will allow in the not too distant future, the increase of the physical safety of drivers and passengers, the optimization of daily traffic, the notification of real-time road congestions, the propagation of alerts of accidents or obstacles on the road, the distribution of useful information for drivers (e.g., nearby restaurants, hotels, gas stations), the access to social networks, file-sharing services, or chats, as well as the connection to external networks such as Internet.

The growing of traffic density on the roads of most towns and cities around the world is becoming a problem. This growing brings traffic congestion on the roads, resulting in negative effects on traveling time, traffic safety, air pollution, noise disturbance, and energy consumption. Therefore, the task of controlling and optimizing the vehicular flows, in agglomerations and their outskirts, is one of the main activities of traffic engineering, seeking to benefit the communities. Before tackling such a complex problem as the optimization of vehicular traffic, the main point is to know how that traffic behaves, i.e., to obtain reliable models of the same. How many vehicles use a road section? Something as simple as that is hard

and expensive to know in most of our main cities. If we lack the number of vehicles on a road, we cannot know or estimate the occupation in a certain road section, or propose a dynamic schedule for the traffic lights at an intersection, etc. However, the actual ATCSs (Adaptive Traffic Control Systems) have been using basic "in situ" technologies (e.g., inductive loops, digital cameras, video cameras, thermal cameras, pneumatic road tubes, magnetic sensors, radars, piezoelectric sensors, infrared beams) to reduce road accidents and optimize traffic flows, which could be dramatically improved with the integration of emerging technologies such as VANET.

As stated before, in "Traffic Engineering", a specific problem to be solved is the development of algorithms to optimize the cycles of traffic lights of a set of intersections and thus achieve a greater vehicular flow with fairer waiting times for all vehicles. Therefore, much research has been done in the field of ATCSs [7] and traffic congestion detection [8][9][10] to improve the flow of vehicles. As we have seen, there is no single solution to solve the above problem. The range of initiatives is wide, and many of them must be applied together to get tangible results. However, to improve existing solutions or to propose new ones, basic algorithms and tools must be developed. An example of such tools is the counting of vehicles with a specific characteristic or within a specified geographical area. According to [11][12][13], there are some alternatives or proposals for counting vehicles based primarily on "in situ" technologies. These "in situ" technologies are complex to install, and they suffer a high economic cost caused by both, installation and recurring maintenance. Due to the huge number of roads worldwide, alternatives to "in situ" technologies must be considered. The VANET technology seems to be a good option for vehicle counting and should become ubiquitous promptly since it is estimated that all vehicles will be equipped with a WAVE device within the next 15 years [14]. In addition, in the case of WAVE, the costs of installing and maintaining the technology are shared between the owners of the vehicles and the organization that maintains the roads (town hall, city hall, county administration, state government, highway administration, etc.) That is, the owner of a vehicle will have to buy an RSU for his/her car, and will have to pay the charges related to its maintenance. Local, national, or international establishments will install and maintain RSUs on the road infrastructure.

In this research work, we propose a novel algorithm to count vehicles that are stopped at a traffic light based on VANET technology, as a basic and integral tool for the development of applications for the ITS. With the aim of validating the proposed algorithm, we use a discrete event network simulator called OMNeT++ in conjunction with a road traffic simulator known as SUMO (Simulation of Urban Mobility), and the Veins (Vehicle in Network Simulation) framework that bidirectionally couples the previously mentioned simulators. We test and analyze our proposal in diverse scenarios, where we vary some parameters such as the number of vehicles, the signal propagation range, the number of lanes, the penetration rate, etc. The simulation results show that our novel algorithm performs an efficient vehicle counting very close to the real one, with a short response time and a small number of control messages.

We have structured the rest of this paper in the following way. First, we review the previous work in Section 2. Then, in section 3, we introduce in details our novel algorithm to count vehicles that are stopped at a traffic light using WAVE technologies. Section 4 justifies our selection of the used simulation tools and briefly describes the testbeds for the validation of the proposed algorithm. A discussion of the results obtained by our simulations is done in Section 5. In the last section, we conclude and give directions for future work.

2 Related Work

Vehicle counting represents a tool that has numerous applications, and due to its usefulness, it has been done in various ways or disciplines with several technologies that makes it applicable to diverse situations. Up to now, most of the proposals to count vehicles, with greater or lesser accuracy, are based mainly on methods or techniques supported by conventional "in-situ" technologies (e.g., inductive loops, digital cameras, video cameras, thermal cameras, pneumatic road tubes, magnetic sensors, radars, piezoelectric sensors, infrared beams) [15].

In the specialized literature, there are many methods, techniques, and algorithms based on the "in situ" technologies mentioned above. For example, there is a lot of work done with images or recordings of digital or video cameras. Chintalacheruvu and Muthukumar [16] proposed an efficient video-based vehicle detection system constructed on top of Harris-Stephen corner detector algorithm [17]. The algorithm was used to develop a standalone vehicle detection and tracking system that determines vehicle count and speed at arterial roadways and freeways. The authors of [18][19] employed images obtained from video cameras to count vehicles in real time. Peiris and Sonnadara [20] used a single digital camera to extract various traffic parameters, including vehicle count, density, and type at a three-way junction.

Sensor networks have also been used to count vehicles. Knaian [21] developed a low-cost package, based on anisotropic magnetoresistive magnetic field sensors that can count passing vehicles. According to the author, the sensors can operate in the roadbed for at least ten years without maintenance, and do not require running wires under the road, facilitating a wide deployment. Litzenberger et al. [22] proposed an embedded system based on an transient optical sensor that is capable of detecting, counting, and measuring the velocity of passing vehicles.

Contreras and Gamess [23] proposed an algorithm to count objects (people, animals, devices, etc.) with wireless technologies (IEEE 802.11) in circular-bounded areas, using several non-overlapping communication channels. In the field of counting vehicles based on VANET technologies, just a few efforts have been made up to today. Gamess and Mahgoub [1] proposed a method to obtain the length of a line of vehicles stopped at a traffic light, by using VANET technology. The algorithm is based on an effective propagation of a request message from the beginning of the line (started by the traffic light) towards the end, and the transmission of the correspondent response message from the last vehicle to the traffic light, using multi-hop, with the expected length.

According to the authors, a possible approximation of the number of vehicles can be obtained by dividing the resulting length by a constant value (e.g., 7 meters), where 7 meters represents the average space to accommodate a vehicle in a line. Unlike the present work, the counting obtained in [1] is an approximation.

Some other works do not count, but estimate the density of vehicles in a specific region. In their work, Luo, Wei, Cheng, and Ren [24] developed an innovative query-response framework which not only enables vehicles to detect the traffic crowdedness of their surrounding region, but also enables vehicles to obtain the remote region traffic crowdedness by sending query messages and fusing reply messages.

Generally speaking, a considerable amount of work has been done with "in-situ" technologies to count vehicles in different scenarios. However, algorithms based on VANET technologies are still very rare, and new proposals are welcome to consolidate this area of knowledge.

3 Algorithm to Count Vehicles that are Stopped at a Traffic Light, using VANET Technologies

In this section, we describe our novel algorithm to count vehicles that are waiting for the traffic light to change from red to green.

3.1 Requirements and Assumptions

Note that in this paper, we use the word "unit" interchangeably with the word "vehicle". They are one and the same. Also, we call "originator" the RSU which initiates the counting process, i.e., the entity that requires the number of vehicles around it, up to a specified range or hop count (called *Hop Limit* in our algorithm). As can be seen, the field *Hop Limit* delimits the counting range, so that application developers will have to select this parameter according to their needs. In our simulations, the RSU starts counting with a value of *Hop Limit* equal to 3, but any value can be used according to the type of applications where the algorithm will be used.

For the implementation of our novel algorithm, we only require the usage of a unique channel, of the seven channels that are available in the DSRC band (5.850–5.925 GHz) [25]. We also assume that each vehicle is capable of determining its actual position on the road using, for example, location services like the Global Positioning System (GPS) [26]. In our work, the location is specified through the latitude and the longitude. However, the same algorithm can be modified to use Cartesian coordinates, by choosing an origin and the direction of the axis. The vehicles that do not have a WAVE device will not be counted, since there is no way to detect them (a penetration rate of 100%). Additionally, the algorithm requires symmetric radio ranges, i.e., there is no one-way communication between two vehicles (if vehicle V_1 can communicate with vehicle V_2, a transmission from V_2 will also reach V_1).

3.2 Structure of Unicast COUNT_REQUEST and COUNT_REPLY Messages

The COUNT_REQUEST and COUNT_REPLY messages are unicast messages propagated by the RSU and the vehicles in the process of counting vehicles stopped at a traffic light. When starting the counting, the RSU will send a unicast COUNT_REQUEST message toward the last vehicle in the line of waiting vehicles, and later this last vehicle will respond with a unicast COUNT_REPLY message that will be transmitted toward the RSU. Both messages have the same Protocol Data Unit (PDU) and are composed of 10 fields (see Fig. 1).

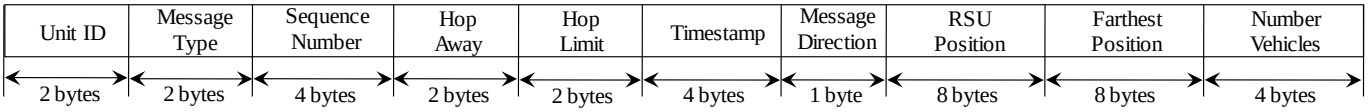

Fig. 1 COUNT_REQUEST and COUNT_REPLY Messages

The field *Unit ID* represents the identification of the sender vehicle or RSU. The value of *Unit ID* must be unique. *Message Type* can be either 0 or 1 and is used to identify the type of message. A value of 0 identifies a COUNT_REQUEST, while 1 is for a COUNT_REPLY. *Sequence Number* is used to match COUNT_REQUEST messages with COUNT_REPLY messages and to distinguish between different requests. The RSU and the vehicles transmit COUNT_REQUEST messages along with the argument *Hop Away* which represents the number of hops-away the receiver of the message is from the RSU. The RSU is the unit that initiates the process of counting specifying a value of *Hop Away* equal to 1. Each vehicle that retransmits the COUNT_REQUEST message shall increment this value by 1. The units will discard the message when the value of *Hop Away* is greater than *Hop Limit*. The field *Hop Limit* is a way to control how far away COUNT_REQUEST messages can be forwarded. It delimits the counting range. *Timestamp* is set by the RSU when it sends the COUNT_REQUEST message. It is a timestamp taken by the RSU at the moment of sending the COUNT_REQUEST message and is aimed to control out-of-date messages and replay attacks. *Message Direction* indicates in which direction the message must be transmitted. For this, the four least significant bits of the *Message Direction* field are used to indicate one of four possible directions (North, South, East, and West). For example, if the message must be transmitted in all directions, then all lowest four bits of the field *Message Direction* must be set to 1 (1111). *RSU Position* is the position (latitude and longitude) of the RSU, and is set by the RSU when sending the COUNT_REQUEST message. *Farthest Position* is the location (latitude and longitude) of the actual known unit that is farthest away from the RSU in the line of vehicles. *Number Vehicles* is set with the number of vehicles counted up to now during the transmission of the

COUNT_REQUEST message. In other words, before the retransmission of a COUNT_REQUEST, a unit must update the value of this field by adding the number of valid neighbors stored on its neighbors list. In the forwarding of COUNT_REPLY messages back to the RSU, its value is not altered.

3.3 Structure of BEACON Messages

The PDU of BEACON messages is composed of 4 fields as depicted in Fig. 2.

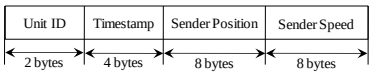

Fig. 2 Structure of a BEACON Message

Unit ID refers to the sender identification. *Timestamp* is the actual time set by the unit when it sends a BEACON message. The synchronization of time between the different units is solved with the time received from the GPS satellites. *Sender Position* and *Sender Speed* represent the actual position (latitude and longitude) and speed of the unit when it sends the BEACON message, respectively.

3.4 Discovery Protocol for Neighboring Vehicles

As stated before, we propose a discovery protocol of 1-hop neighbors that helps in the transmission of both the COUNT_REQUEST messages started by the RSU and the COUNT_REPLY messages started by the last vehicle, in the opposite direction.

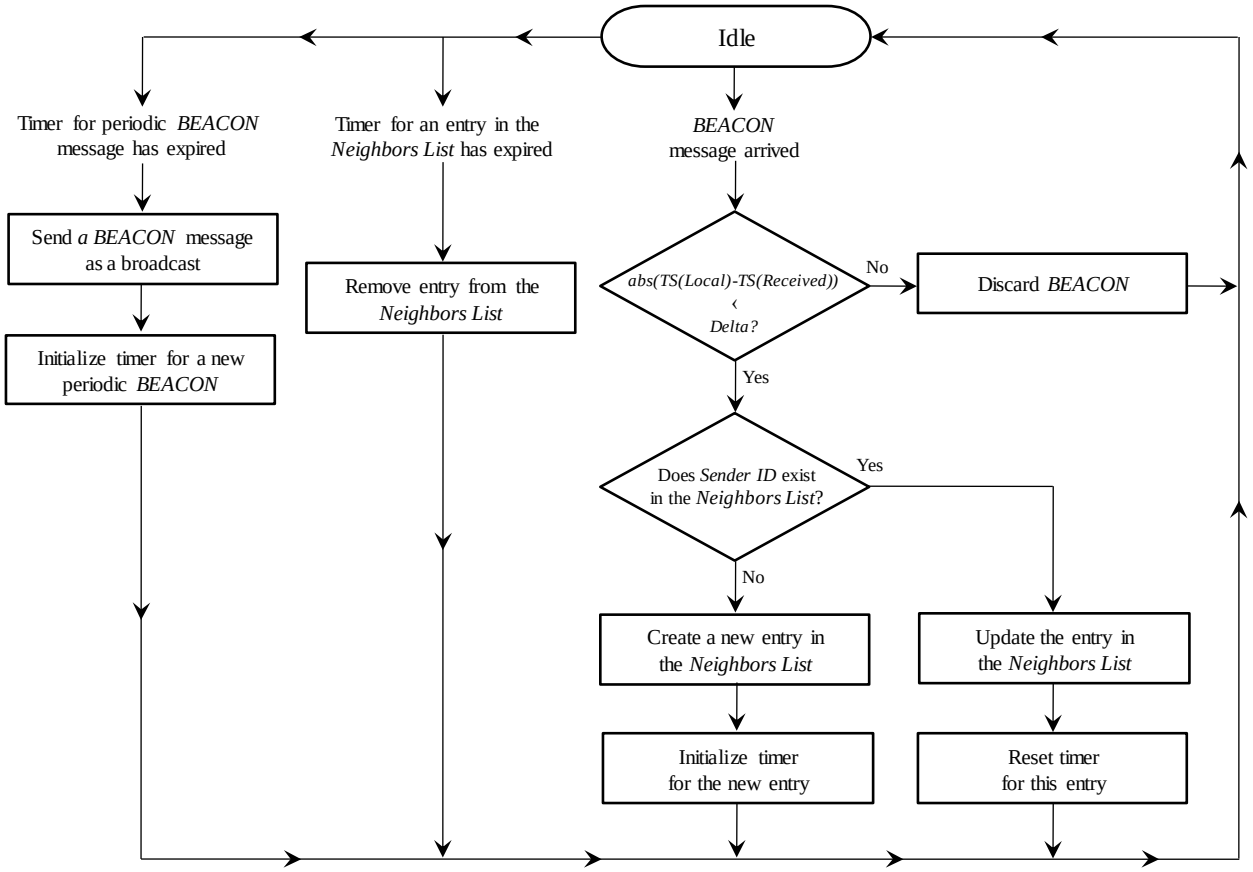

Fig. 3 Flow Diagram for Updating the Neighbors List

Our algorithm uses BEACON messages to periodically exchange the necessary information between any two in-range neighbors to maintain a list of 1-hop neighbors. The number of neighboring vehicles around one unit can be easily obtained from its neighbors list. Every unit periodically broadcasts BEACON messages that include its *Unit ID*, a timestamp, and its actual position and speed (see Fig. 2), so that, 1-hop neighbors are aware of its presence, position, and speed. Position and speed are obtained by units from their GPS receivers. When a vehicle receives a BEACON message, it first checks the *Timestamp* field (see Fig. 2) to validate that the BEACON message is current and not a copy of a previous message injected by a replay attack. If the *Timestamp* is valid, then the unit checks whether or not the *Unit ID* exists in its list of 1-hop neighbors. If the *Unit ID* does not exist, a new entry is created and the information of this neighbor is stored. Otherwise, the information of the fields *Sender Position* and *Sender Speed* for the sending vehicle are just updated as well as the associated timer. With this information, the unit can interpolate the actual position of its 1-hop neighbors at any time. Moreover, entries in the neighbors list that are not updated during a certain period of time will be considered stale and then removed. The flow diagram for creating a 1-hop neighbors list using BEACON messages is given in Fig. 3. The BEACON interval is set to 1s to ensure that the information in the neighbors list is always up-to-date.

3.5 Algorithm

Beside of the neighbor discovery protocol described before, the basic approach of the algorithm is:

1) Propagate a unicast COUNT_REQUEST, from the RSU toward the vehicle that is farther away in the line of vehicles, with the total number of vehicles counted up to now (called *Number Vehicles* in our algorithm)
2) Propagate a unicast COUNT_REPLY in the opposite direction, i.e., from the last vehicle in the line toward the RSU, with the total number of vehicles calculated according to the propagation of the previous COUNT_REQUEST message.

Fig. 4 depicts a simplified flow diagram for the procedure followed by the RSU in the counting algorithm. The RSU starts the vehicle counting by sending a unicast COUNT_REQUEST message (see Fig. 1) with its own geographic location in the field *RSU Position*, toward the last unit in the line of waiting vehicles. For this, the RSU will determine and put in the *Farthest Position* field the location (latitude and longitude) of the vehicle that is farther away from it in the line and within its propagation range. The RSU will also set in the field *Number Vehicles* the result of computing the total number of vehicles in its neighbors list, that are waiting in the line. Additionally, the RSU will specify a value of *Hop Away* equal to 1 and put a time sample in the *Timestamp* field.

Now, when the RSU receives a COUNT_REPLY message, it will first validate its *Timestamp* field. If the timestamp is not within the expected interval of time, the COUNT_REPLY is discarded. Otherwise, the RSU will obtain the total number of vehicles in the field *Number Vehicles*.

It is worth to point out that not all the entries that are in the neighbors list of the RSU are valid for the counting. That is, the neighbors list also includes vehicles that are moving in the opposite direction and vehicles that have already passed the traffic light. However, the vehicles in the reverse direction can be easily discarded in accordance to their speed. Also, the actual location can be used to distinguish vehicles that have already passed the traffic light.

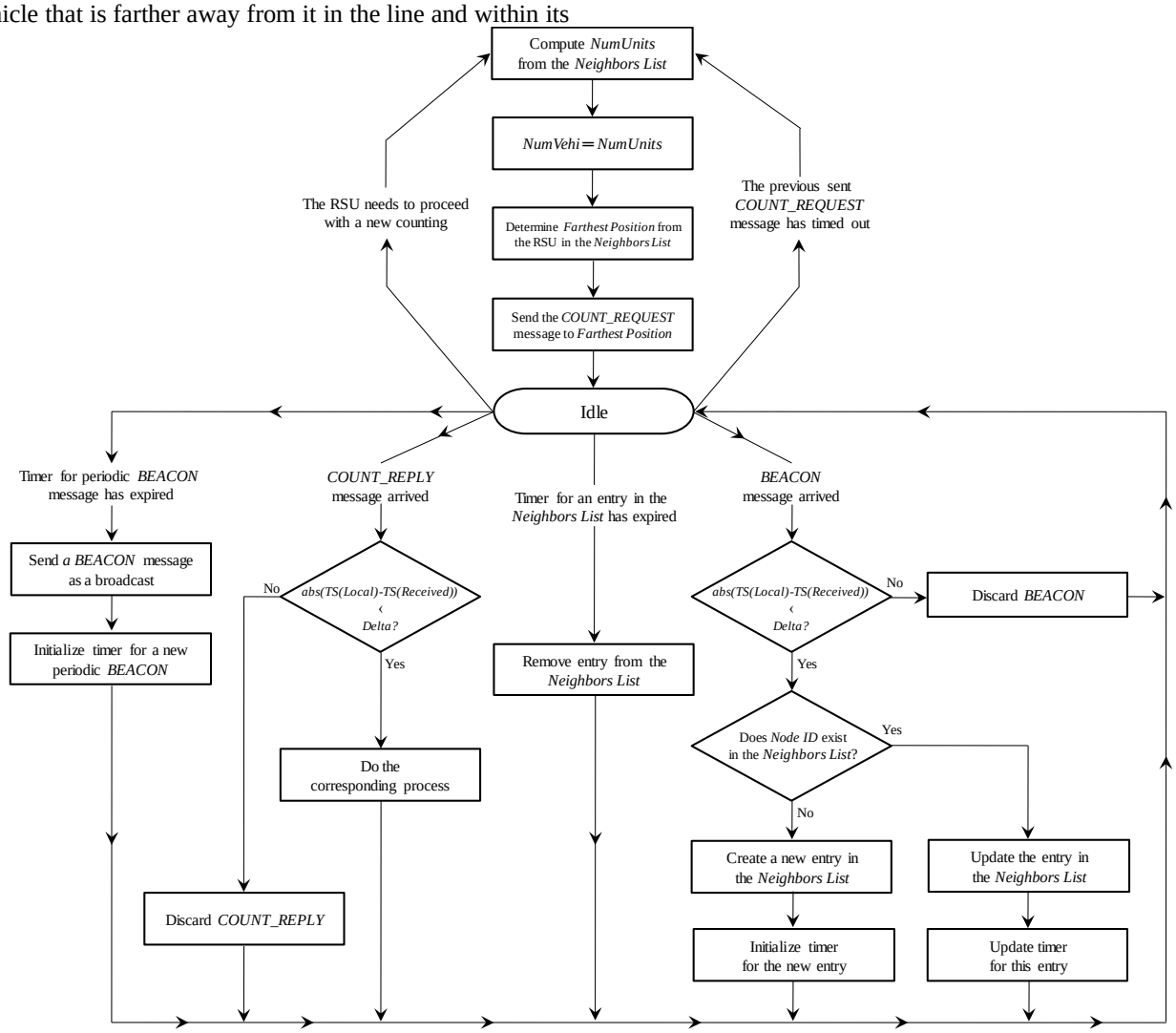

Fig. 4 Flow Diagram of the Procedure Followed by the RSU to Count Vehicles Stopped at a Traffic Light

Fig. 5 depicts a simplified flow diagram for the procedure followed by the vehicles in the counting algorithm. When a vehicle receives a COUNT_REQUEST message, it will first validate the fields *Timestamp* and *Hop Away*. If the timestamp is not within the expected interval of time or the number of hops has been exceeded, the COUNT_REQUEST is discarded. Otherwise, the behavior of the vehicle will depend on whether or not it is the last unit of the line, or whether or not the field *Hop Away* is equal to the field *Hop Limit*. If so, the vehicle will start to send a unicast COUNT_REPLY message back to the RSU. As can be appreciated, only this "last unit" in the line is responsible for eliminating the COUNT_REQUEST and replacing it by a COUNT_REPLY that moves in the opposite direction. It is a copy of the COUNT_REQUEST message with *Message Type* equal to 1 (to indicate a COUNT_REPLY) and a *Unit ID* field updated to the correct ID. Note that the value of *Hop Away* and *Farthest Position* are not modified during the propagation of the COUNT_REPLY, allowing the RSU to know how far away (in hops and in meters), the counting was done. Otherwise, if the vehicle is not the "last one" of the line, then it will make the following modifications: (1) increment by 1 the *Hop Away* field, (2) update the field *Number Vehicles* based on the information from its neighbors list, (3) determine the farthest vehicle from the RSU in the line within its range, and (4) resend the COUNT_REQUEST message to the unit defined in the *Farthest Position* field.

Now, when a vehicle receives a COUNT_REPLY message, it will first validate its *Timestamp* field. If the timestamp is not within the expected interval of time, the COUNT_REPLY is discarded. Otherwise, the vehicle will determine from its neighbors list the closest unit to the RSU (the next forwarder) in the line, within its propagation range, and will resend the COUNT_REPLY message to this unit. It is obvious that if the originating RSU is within the propagation range of the vehicle, the COUNT_REPLY message will be sent to it directly.

New units can be added in the line of waiting vehicles at any moment. If an RSU had already made a counting before the arrival of new vehicles, this RSU would not be informed about the change, unless it starts a new counting. That is, the proposed algorithm is based on the request/response model, and the counting obtained only applies for a specific time.

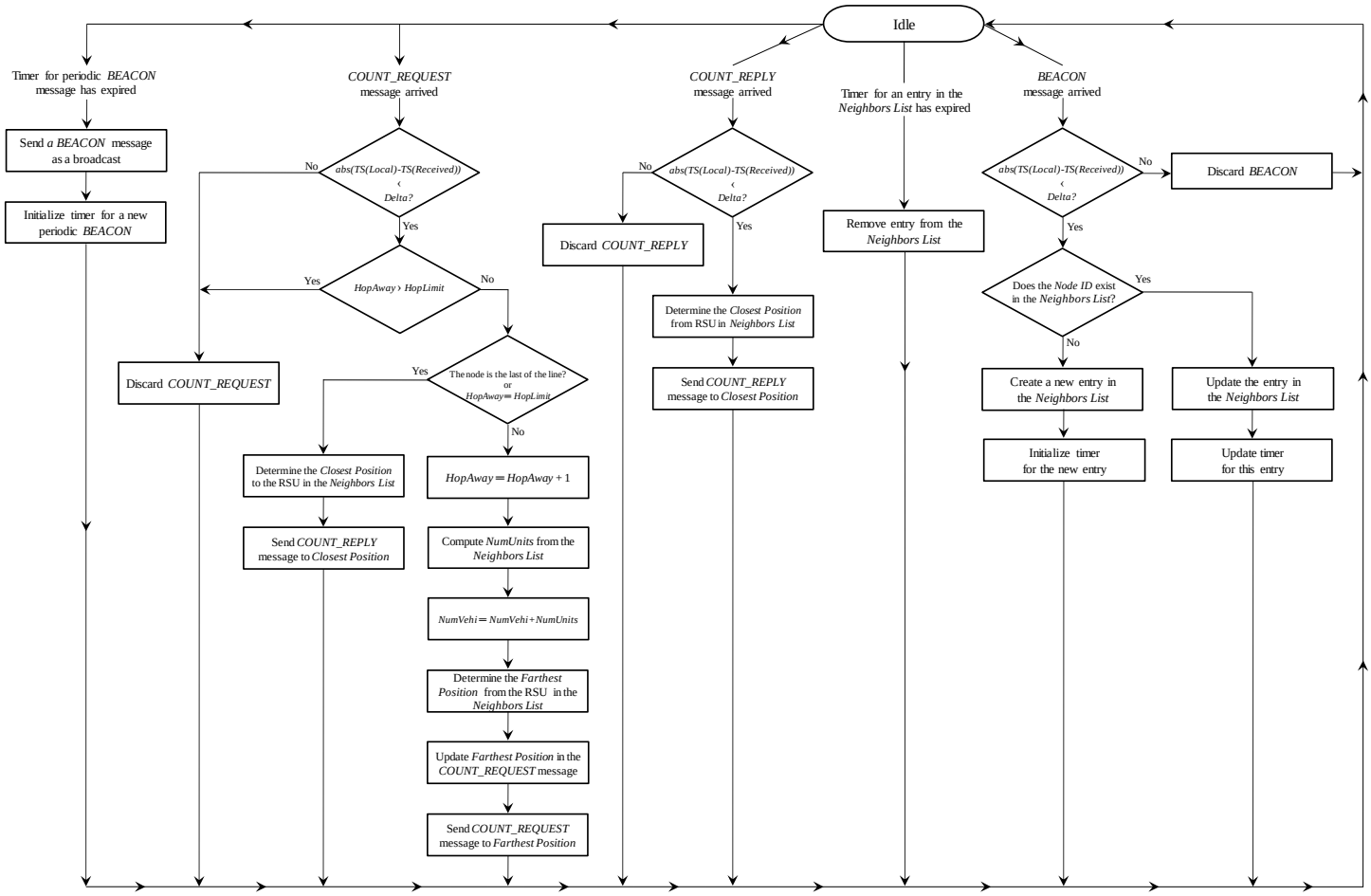

Fig. 5 Flow Diagram of the Procedure Executed by Vehicles to Count Units Stopped at a Traffic Light

3.6 Example of Propagation

In this section, we present an example of the propagation of a COUNT_REQUEST (Fig. 6) and a COUNT_REPLY (Fig. 7) in a scenario with one lane and 18 vehicles, where V_1, V_2, V_3, …, V_{18}, are the vehicles in the line that are stopped at a traffic light waiting for the green light. Circles around the units represent the propagation range of messages (COUNT_REQUEST, COUNT_REPLY, and BEACON) sent

by the units. To facilitate the explanation of the example, we will assume that the radius of the propagation range of a message is equivalent to five vehicles. In a real scenario, it will be bigger than five vehicles since the range of DSRC is targeted to be up to 1 km [1]. The basic operation of the algorithm is as follows: Before the initiation of the counting process, the RSU will listen to BEACON messages to discover vehicles that are within its propagation range. In this case, the RSU will detect the presence of V_1, V_2, V_3, V_4, and V_5 that are waiting for the green light, where V_5 is the farthest away vehicle from the RSU. Therefore, the RSU will transmit a unicast COUNT_REQUEST message to V_5 (see Fig. 6) with *Hop Away* equal to 1, *Farthest Position* with the location of V_5 (the next forwarder), and *Number Vehicles* set to 5. Using the information from its neighbors list, V_5 determines that there are five vehicles (V_6, V_7, V_8, V_9, and V_{10}) that are waiting in the line and are farther away from the position specified in the field *Farthest Position* of the received COUNT_REQUEST. So, vehicle V_5 resends the COUNT_REQUEST to V_{10} with the appropriate changes by incrementing by 1 the value *Hop Away* (its new value is 2), setting the location of V_{10} in the *Farthest Position* field (the farthest unit from the RSU discovered by V_5), and adding 5 to *Number Vehicles* (its new value is 10). The process of counting will continue with the transmission of the COUNT_REQUEST by vehicle V_{10}, followed by vehicle V_{15} (see Fig. 6). In this case, vehicle V_{10} will resend the COUNT_REQUEST message with *Hop Away* equal to 3, *Farthest Position* set to the location of vehicle V_{15}, and *Number Vehicles* equal to 15, whereas V_{15} will resend the COUNT_REQUEST message with *Hop Away* equal to 4, *Farthest Position* set to the location of vehicle V_{18}, and *Number Vehicles* equal to 18. Vehicle V_{18} will know that it is the last vehicle in the line according to information from its neighbors list. Hence, V_{18} will start the process of the propagation of the COUNT_REPLY message (with *Number Vehicles* equal to 18) back to the RSU (see Fig. 7).

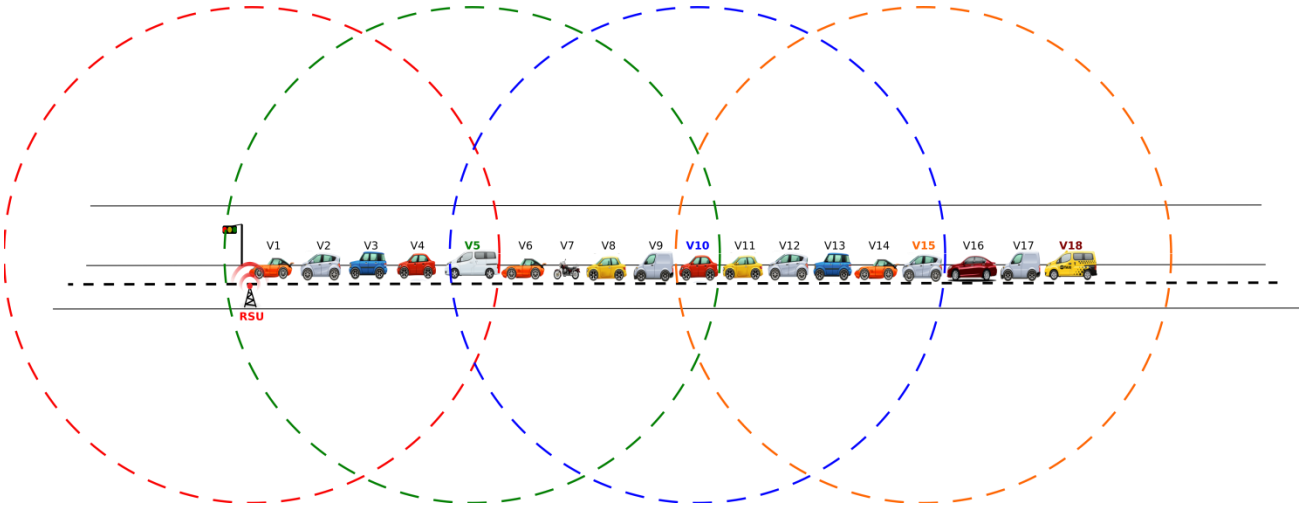

Fig. 6 Propagation of the COUNT_REQUEST Message

We can observe in Fig. 7 that vehicle V_{18} initiates the process of the propagation of the unicast COUNT_REPLY message back to the RSU, by sending it to V_{13}, the closest unit to the RSU discovered by V_{18}. This message is a copy of the COUNT_REQUEST received with small changes (*Unit ID* and *Message Type* are the only modified fields). That is, the following important fields will be kept unchanged: *Hop Away* equal to 4, *Farthest Position* set to the location of vehicle V_{18}, and *Number Vehicles* equal to 18. When vehicle V_{13} receives the COUNT_REPLY, it will resent it to V_8 according to the result of the selection of the closest unit to the RSU from its neighbors list. The above process will continue in sequence with the retransmission of the COUNT_REPLY message by vehicles V_8 and V_3, up to the RSU. Finally, when the RSU receives the COUNT_REPLY from V_3, it simply processes the results obtained in the PDU.

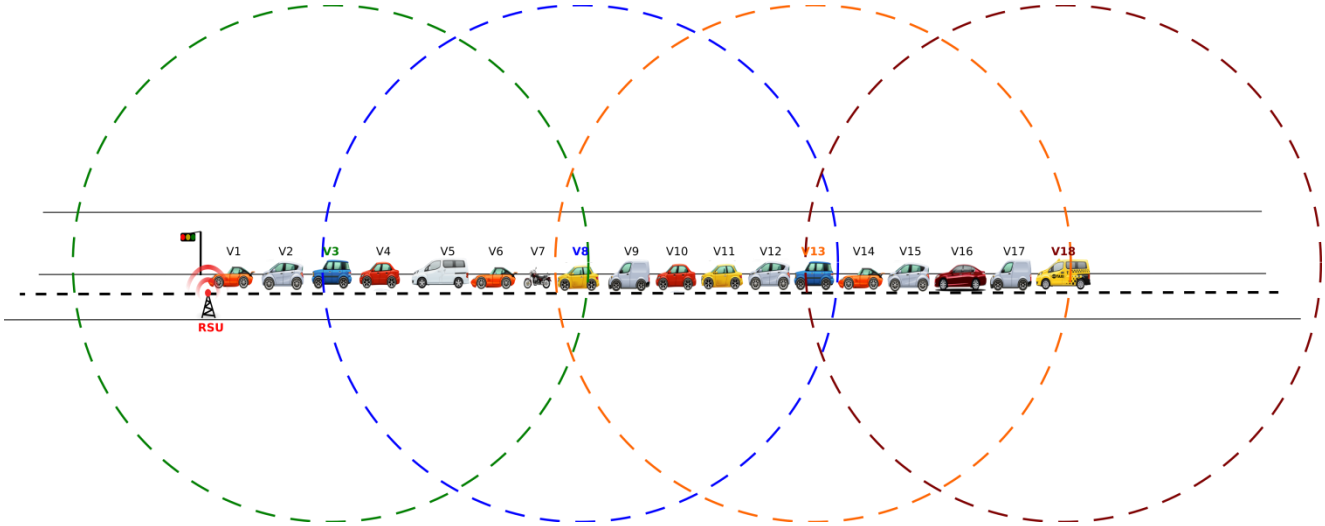

Fig. 7 Propagation of the COUNT_REPLY Message back to the RSU

4 Environments and Scenarios for Simulation

To evaluate the accuracy and performance of our novel algorithm, we carried out extensive simulation experiments with different sets of parameters. This section aims to present the selected simulation tools and common parameters for this evaluation.

4.1 Simulation Tools

Nowadays, there are numerous simulation tools ranging from open source to commercial products. In any research work, it is always important to choose the most appropriate. A comprehensive study about current simulators, their characteristics, capabilities, and approaches is provided in [27].

Up to now, there are no simulation tools that cover vehicle mobility and networking, at the same time. That is, on the one hand, vehicle mobility simulation tools have been proposed for researchers in the field of traffic engineering. The objective of these tools is to import road networks from well-known maps (e.g., Google Maps or OpenStreetMap) and to generate realistic vehicular traffic flows over the roads, by specifying some constraints. On the other hand, networking simulators with very basic mobility models are used by researchers in the area of networking. For traffic engineering, some open-source projects have been actively used by the community, such as VanetMobiSim and Simulation of Urban MObility (SUMO) [28]. Unfortunately, VanetMobiSim seems to be a dead project now. Its last version (version 1.1) was released in February 2007. SUMO is an open source, highly portable, microscopic and continuous road traffic simulation package designed to handle large road networks. For network simulation, two open-source simulators outstand (ns-3 and OMNeT++). OMNeT++ [29] is an open-source, multiplatform (Windows, MacOS, and Linux), C++ based discrete event simulator for networking. Through its GUI, users can create topology files and inspect the state of each component during simulations [30].

To bridge the gap between the two worlds, some projects propose a way to couple a road traffic simulator with a network simulator, which seems to be the only viable solution in the present time to do VANET simulations. For our work, we used an open source bidirectional simulation framework called Vehicles in Network Simulation (Veins) [31]. Veins couples SUMO with OMNeT++ using the Traffic Control Interface (TraCI) [32]. Veins already implements the WAVE protocol stacks. It is most noticeable for IEEE 802.11p, IEEE 1609.4 multi-channel operation, and comprehensive models for the MAC and PHY layers. We implemented the algorithm on top of WAVE Short Message Protocol (WSMP) and IEEE 802.11p. Unlike the standard IP protocol, WSMP allows applications to directly control the lower-layer parameters such as transmission power, data rate, channel number, and receiver MAC addresses.

We chose the Veins framework because it includes a complete suite of models to make vehicular network simulations as realistic as possible, without sacrificing the speed of execution. Additionally, Veins offers interesting features such as online reconfiguration and re-routing of vehicles in reaction to the network simulator.

We simulated different scenarios where vehicles are stopped at a traffic light. Table 1 summarizes the technical parameters shared by all the scenarios and simulated cases of our algorithm. For all our simulations, we selected WAVE (IEEE 802.11p) for the wireless communication standard, with a bitrate of 18 Mbps. The propagation model used in the simulations was the two-ray ground model. We opted for this model because it is suitable for predicting signal strength over distances of several kilometers, so for a vehicular network where distances can be long, it gives better results in terms of accuracy compared with other models. It is worth to note that the bitrate, modulation and coding were chosen based on [33].

Table 1 Simulation Parameters

Parameter	Value
Type of roads	Main roads
Length of Road Section	8 km
Wireless Standard	IEEE 802.11p

Transmission Bitrate	18 Mbps
Transmission Power	20 mW (13 dBm)
Receptor Sensitivity	-89 dBm
Thermal Noise	-110 dBm
Message Type	WSMP data
Channel Bandwidth	10 MHz
Frequency Band	5.850–5.925 GHz
Radio Propagation Model	Two-ray ground
Simulation Time	120 seconds
Vehicle Beacon Interval	1 second

Another important point to consider is the definition of a stopped vehicle at a traffic light. There is no doubt that it is a complex topic, and it will vary from person-to-person. In our simulations, we considered two cases. The first case is related to vehicles that are close-by the RSU (at a distance inferior to 15 meters). These vehicles are considered stopped at the traffic light if the light has been red for at least 0.5 seconds, and the vehicles are immobile. For the vehicles that are at a distance of 15 meters or more from the RSU, we considered that they were stopped if they were at a distance of 5 meters or less of another stopped vehicle, and with a speed inferior or equal to 5 km/h.

5 Analysis of the Performance Results of our Simulations

This section discusses the results obtained for our experiments in different scenarios.

In order to evaluate the accuracy and performance of the proposed algorithm, we present and analyze some of the results of our simulations that were executed mainly in three types of scenarios: (1) scenarios with a one-way road of a single lane, (2) scenarios with a one-way road of two lanes, and (3) scenarios with a one-way road of three lanes. The one-way road was 8000m long, and each lane was 3.6m in width (note that 'm' as a unit notation corresponds to meters). We used SUMO [28] to generate the vehicular movement patterns, where the vehicles move with random speeds within the given speed limit of the roads, and according to the vehicles ahead. We run our simulations with different numbers of vehicles.

We consider scenarios where vehicles are equally injected in time into the scenarios (entering the road every 0.5 seconds) in an extremity and move toward the traffic light. The vehicles move forward when the traffic light is green and slow down and wait at red traffic light until the light turns green. Thus, vehicles tend to form a line of vehicles where some of them are stopped by the red traffic light, and others are moving toward it.

We also considered a fourth scenario, where we simulate a simple signalized intersection with two roads that cross each other at right angles (see Fig. 14). We propose this fourth scenario to study the impact of two-way traffic over the algorithm, and see if it presents scalability issues. Finally, in the fifth scenario, we study the impact of the penetration rate of WAVE over the algorithm.

5.1 Scenarios with a One-way Road of One Lane

In this section, we study the accuracy and performance of the algorithm in terms of the number of vehicles counted, the associated response time to count the vehicles, and the number of control messages (COUNT_REQUEST and COUNT_REPLY) sent by the units during the counting, when we vary the number of units and their propagation range in a road with one lane. For all these scenarios, the RSU initiates the counting process with a value of *Hop Limit* = 3.

Table 2 shows the number of units counted by our algorithm when we varied the number of units (25, 50, 75, 100, 125, 150, and 175) and their propagation range (200m, 250m, 300m, 350m, and 400m). The results are represented as values *a*/*b*, where *a* indicates the number of vehicles that are within the scope of the RSU using multihop routing (i.e., vehicles that should be counted) and *b* the number of vehicles actually counted by our algorithm. For these experiments, we can see that the algorithm has a high accuracy in the vehicle counting, specifically for values of the propagation range equal to 300m, 350m, and 400m, even in conditions of high vehicular flow, with an effectiveness between 97% and 100%, in the counting. For example, for a number of units equal to 175 and a propagation range of 300m, the RSU should count 175 units. As shown in Table 2, our algorithm also counted 175 units, being 100% effective in this case. There are some factors that can contribute to the small error observed. The major is due to missing information in the neighbors list of some vehicles, and can be the result of BEACON messages that suffer collisions or BEACON messages not sent on time.

Fig. 8 shows the performance of the algorithm for the response time when we varied the number of units (25, 50, 75, 100, 125, 150, and 175) and their propagation range (200m, 250m, 300m, 350m, and 400m). For each number of units, the results are shown in groups of five bars according to the propagation range value, i.e., the first blue bar corresponds to 200m, the second cyan bar to 250m, the third purple bar to 300m, the fourth green bar to 350m, and the fifth yellow bar to 400m. Each of these bars represents the response time in milliseconds (ms) for the proposed algorithm. We can note that for any number of units, the response time is much lower for higher values of the propagation range, specifically for values equal to 350m and 400m. For example, for a number of units equal to 100, the response time is equal to 24.98ms and 12.65ms for a propagation range of 250m and 400m, respectively.

Table 2 Units Counted when Varying the Number of Units and Propagation Range (Road with One Lane)

Total Number of Units	**Propagation Range Values in Meters**				
	200m	**250m**	**300m**	**350m**	**400m**
25	25/25	25/25	25/25	25/25	25/25
50	50/50	50/50	50/50	50/50	50/50
75	75/73	75/73	75/73	75/74	75/75
100	100/98	100/99	100/100	100/100	100/100
125	125/123	125/124	125/125	125/125	125/125
150	150/149	150/149	150/150	150/150	150/150
175	175/173	175/174	175/175	175/175	175/175

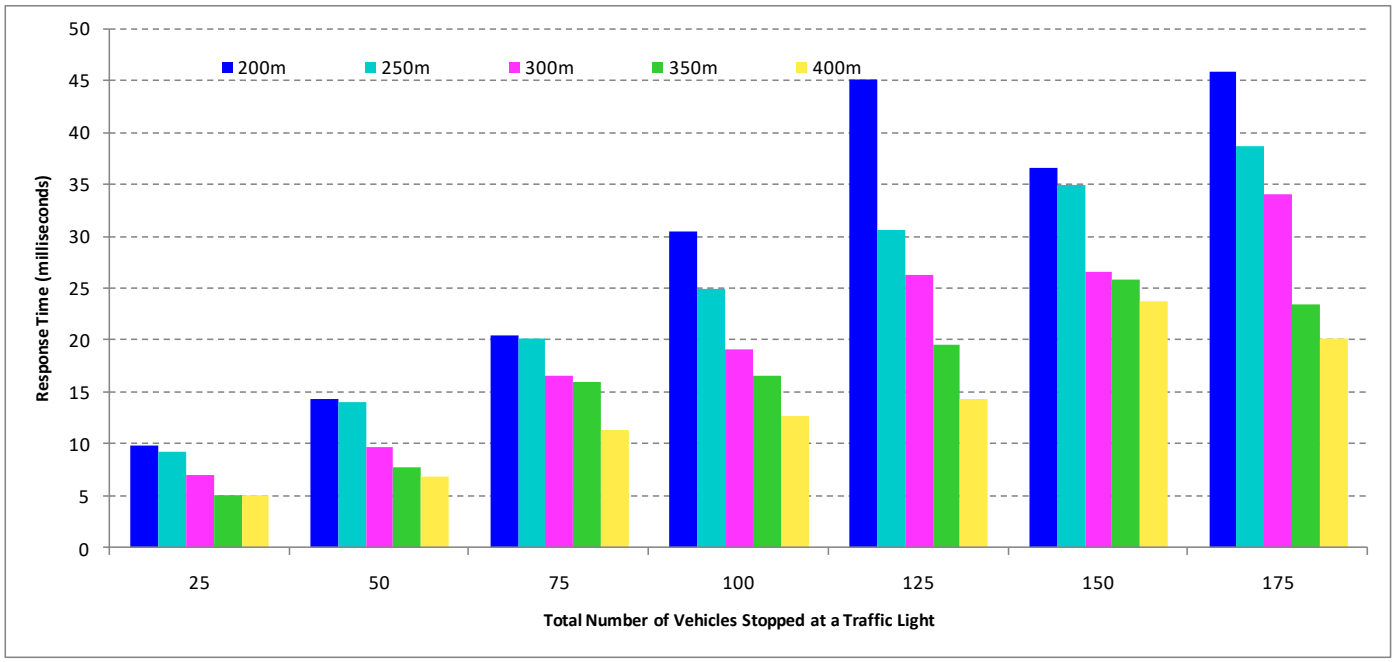

Fig. 8 Response Time in Different Scenarios during the Vehicle Counting (Road with One Lane)

Fig. 9 illustrates the behavior of the algorithm in terms of the total number of COUNT_REQUEST and COUNT_REPLY control messages transmitted by units during the counting of vehicles, when we varied the number of units (25, 50, 75, 100, 125, 150, and 175) and their propagation range (200m, 250m, 300m, 350m, and 400m). We can see that for a given number of vehicles, as we increase the propagation range, the total number of control messages is reduced significantly. For example, for a number of units equal to 175, the number of control messages transmitted is equal to 16 and 10 for a propagation range of 200m and 400m, respectively.

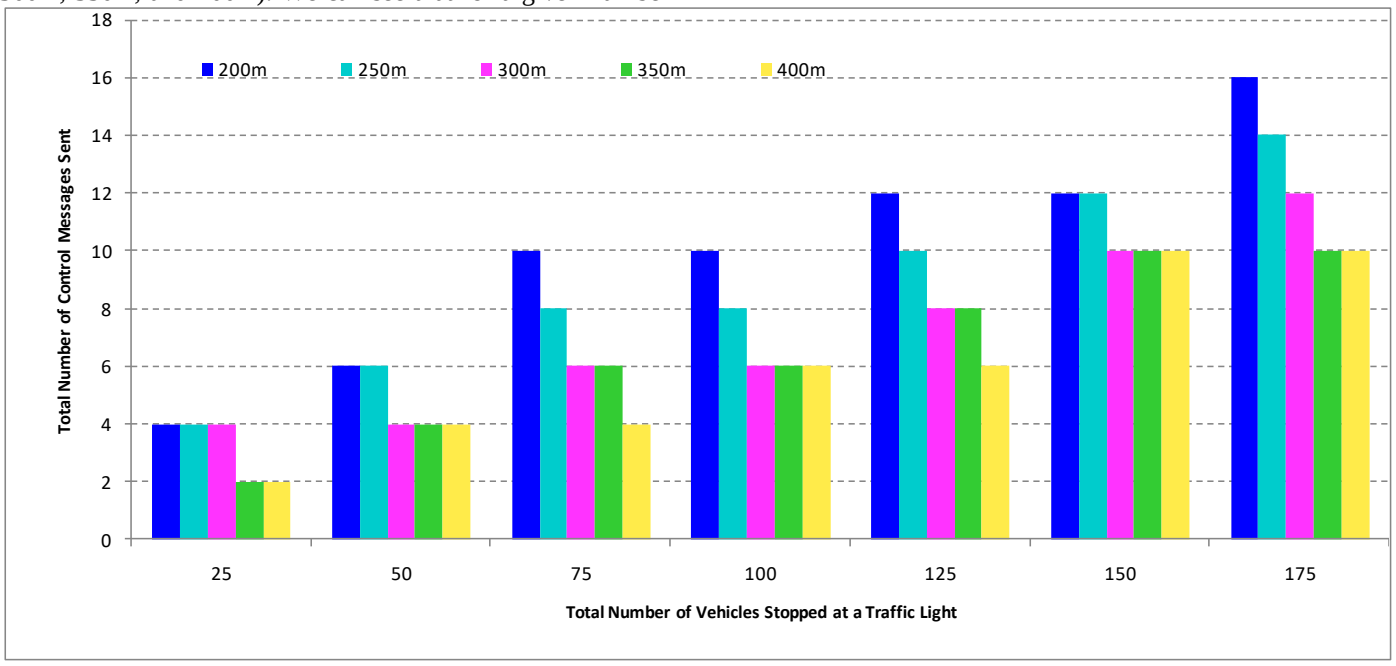

Fig. 9 Total Number of Control Messages Sent in Different Scenarios during the Vehicle Counting (Road with One Lane)

5.2 Scenarios with a One-way Road of Two Lanes

In this section, we study the accuracy and performance of the proposed algorithm in terms of the number of vehicles counted, the response time, and the total number of control messages sent by the units during the counting, in scenarios where the vehicles are stopped at a traffic light on a one-way road with two lanes. At the beginning of the simulations, the vehicles

are distributed in both lanes with a total number of units varying from 50 to 350. The RSU starts the counting process with *Hop Limit*=3.

Table 3 contains the results that we obtained in the simulations concerning the number of units counted. The results are presented as values *a*/*b*, where *a* is the number of vehicles that are within the scope of the RSU using multihop routing (i.e., vehicles that should be counted), and *b* is the number of vehicles actually counted by our novel algorithm. As it can be inferred from Table 3, our algorithm does well in this scenario, and has a high accuracy in the vehicles counting. For example, for a total number of 300 units and a propagation range of 350m, the RSU should count 300 units. In our simulations, our proposed algorithm counted 293 units, resulting in a small error of 2.3%.

Table 3 Units Counted when Varying the Number of Units and Propagation Range (Road with Two Lanes)

Total Number of Units	Propagation Range Values in Meters				
	200m	**250m**	**300m**	**350m**	**400m**
50	50/50	50/50	50/50	50/50	50/50
100	100/99	100/100	100/100	100/100	100/100
150	150/147	150/147	150/147	150/147	150/148
200	200/194	200/194	200/194	200/194	200/197
250	250/242	250/242	250/244	250/244	250/246
300	300/289	300/292	300/293	300/293	300/297
350	350/323	350/327	350/329	350/331	350/334

In Figs. 10 and 11, we varied the total number of units (50, 100, 150, 200, 250, 300, and 350) and their propagation range (200m, 250m, 300m, 350m, and 400m) with the aim of evaluating the behavior of the algorithm with respect to the response time and the total number of control messages sent by the vehicles during the counting, respectively. The RSU started the counting process with *Hop Limit*=3. According to our simulations, the best results are obtained with values of the propagation range of 350m and 400m. For example, for 300 units, we can see that the response time is 36.78ms (see Fig. 10) and the total number of control messages sent is 12 (see Fig. 11) for a propagation range equal to 250m; while it is 25.86ms (see Fig. 10) and 8 messages (see Fig. 11) for a propagation range of 350m. This behavior can be explained by the fact that for a bigger propagation range and a queue length that can be totally covered by the specified *Hop Limit*, the number of actual hops to complete the counting is smaller. As a result, the response time and the number of control messages sent are smaller.

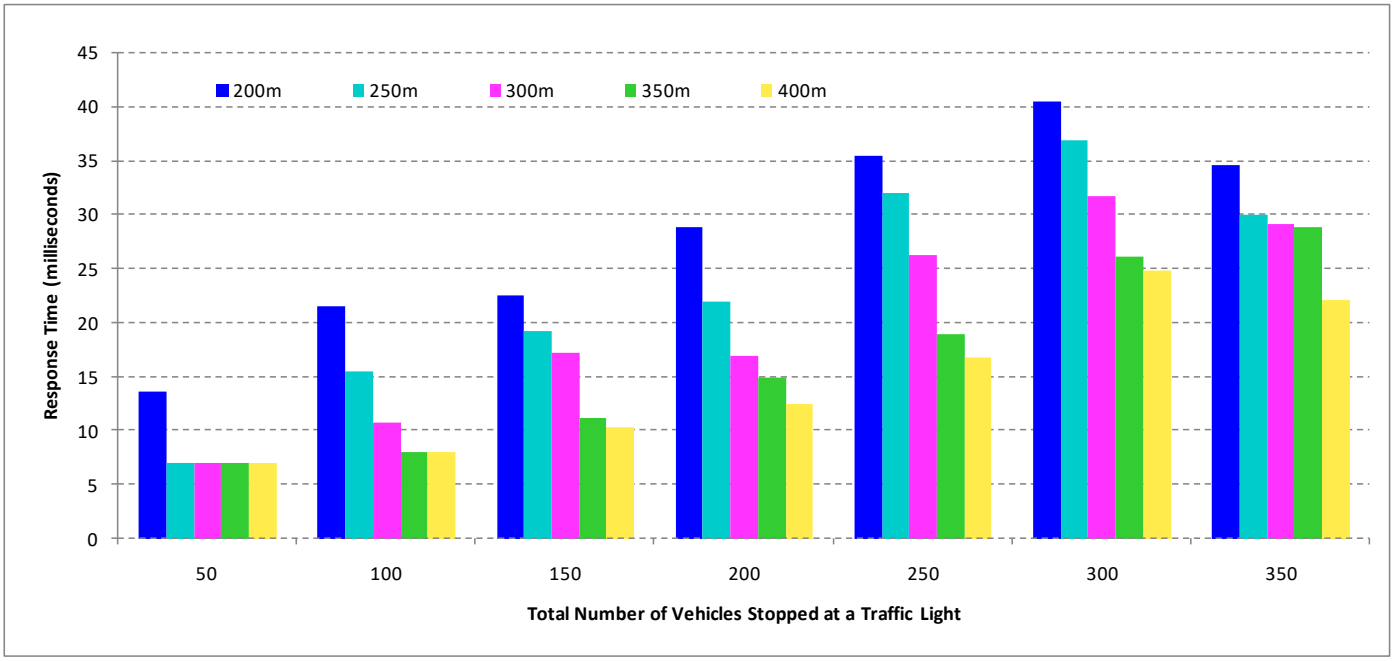

Fig. 10 Response Time in Different Scenarios during the Vehicle Counting (Road with Two Lanes)

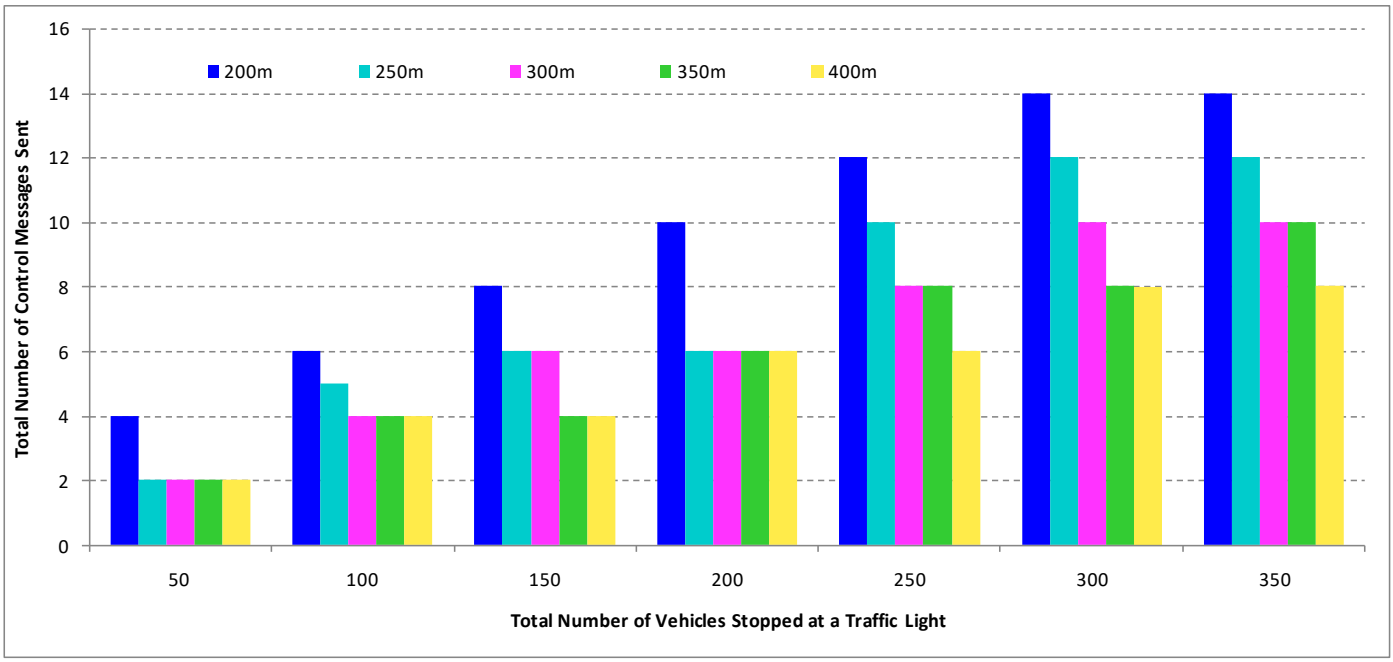

Fig. 11 Total Number of Control Messages Sent in Different Scenarios during the Vehicle Counting (Road with Two Lanes)

5.3 Scenarios with a One-way Road of Three Lanes

In this section, we look at the behavior of the proposed algorithm, in terms of the number of units counted, the response time, and the total number of control messages sent by the units during the counting in scenarios where the vehicles are stopped at a traffic light on a one-way road with three lanes. At the beginning of the simulations, the vehicles are distributed in the three lanes with a total number of units varying from 100 to 400. The RSU starts the counting process with *Hop Limit*=3.

Table 4 shows the results of the experiments for scenarios where we varied the total number of units (100, 150, 200, 250, 300, 350, and 400) and their propagation range (200m, 250m, 300m, 350m, and 400m). Similarly to the experiments of Tables 2 and 3, our algorithm also has a high precision in the counting of vehicles for these scenarios.

Table 4 Units Counted when Varying the Number of Units and Propagation Range (Road with Three Lanes)

Total Number of Units	Propagation Range Values in Meters				
	200m	**250m**	**300m**	**350m**	**400m**
100	100/100	100/100	100/100	100/100	100/100
150	150/147	150/147	150/147	150/150	150/150
200	200/193	200/196	200/196	200/197	200/200
250	250/236	250/242	250/243	250/244	250/246
300	300/289	300/293	300/294	300/294	300/296
350	350/325	350/327	350/330	350/330	350/341
400	400/375	400/380	400/384	400/386	400/390

Figs. 12 and 13 show the results of the simulations for scenarios where we varied the total number of units (100, 150, 200, 250, 300, 350, and 400) and their propagation range (200m, 250m, 300m, 350m, and 400m) with the aim of evaluating the behavior of the algorithm with respect to the response time and the total number of control messages sent by the vehicles during the counting, respectively. In all the simulations, the RSU started the counting process with *Hop Limit*=3. Again, the results of the simulations show good response times with a small number of control messages sent by the units during the counting of vehicles. Also, it is important to mention that the best results are obtained for propagation range values equal to 350m and 400m. For example, for 400 units, we can see that the response time is 34.57ms (see Fig. 12) and the number of control messages sent is 12 (see Fig. 13) for a propagation range equal to 200m; while it is 20.26ms (see Fig. 12) and 6 messages (see Fig. 13) for a propagation range of 400m.

We can see that the results obtained by the algorithm in terms of counting were much more accurate in the scenarios with a one-way road of a single lane (with a accuracy that varies from 97.3% to 100%) compared to those obtained with two (with an accuracy of 92.3% to 100%) and three lanes (with a precision that fluctuates from 92.9% to 100%). However, the response times and the total number of control messages sent by the units during the counting were lower in the scenarios with a one-way road of three lanes.

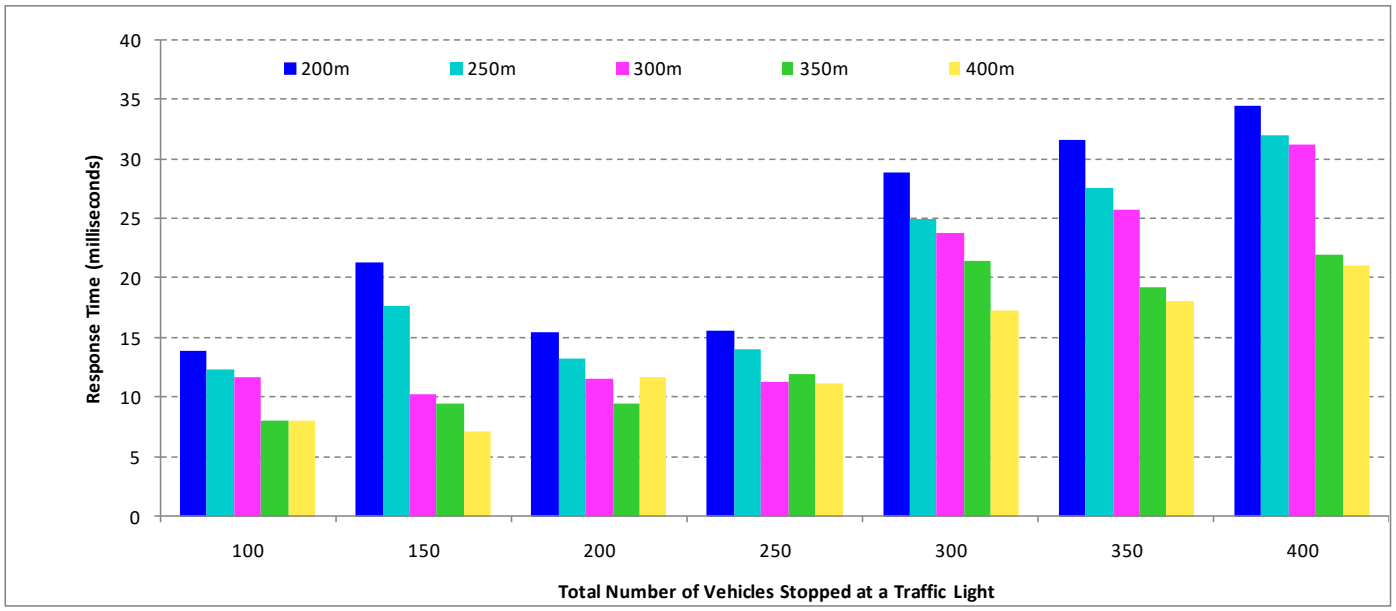

Fig. 12 Response Time in Different Scenarios during the Vehicle Counting (Road with Three Lanes)

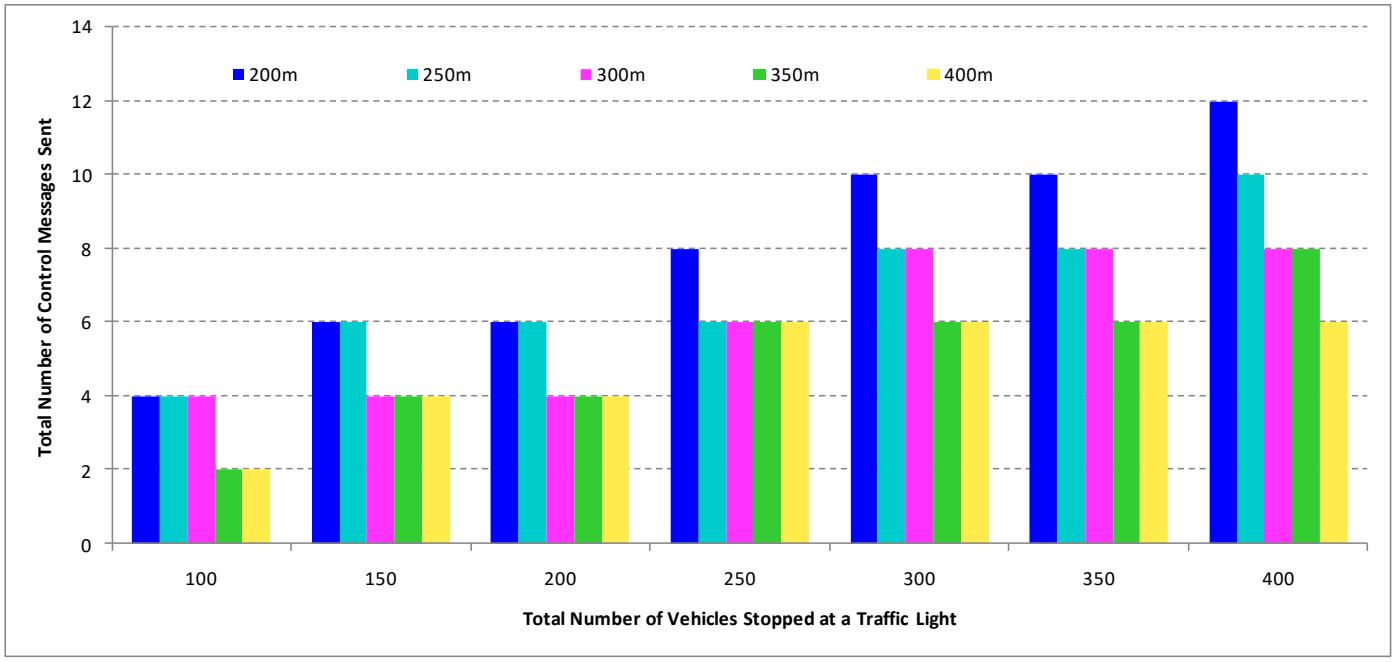

Fig. 13 Total Number of Control Messages Sent in Different Scenarios during the Vehicle Counting (Road with Three Lanes)

5.4 Application of the Proposed Algorithm in a Scenario with a Four-way Intersection

This section deals with the importance of estimating the number of vehicles stopped at a traffic light in road intersections, to improve vehicular traffic. For that, we study the accuracy and performance of our algorithm using a four-way intersection with multiple lanes on both sides (as shown in Fig. 14).

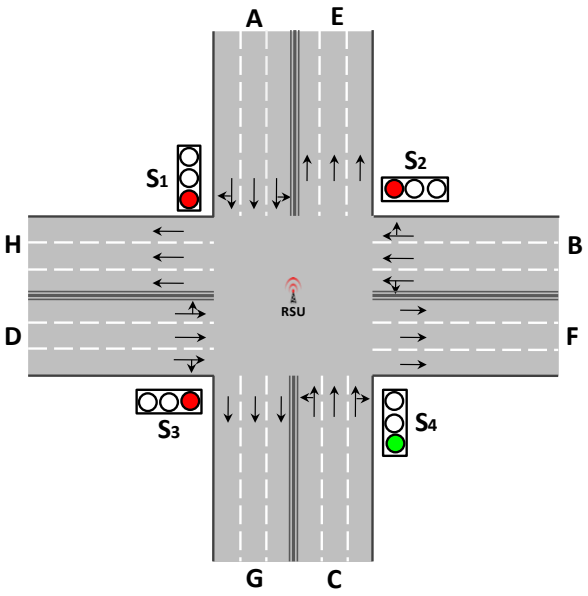

Fig. 14 Outline of a Four-way Intersection

In Fig. 14, labels A, B, C, D, E, F, G, and H indicate the segments of the roads. In the corners of the intersection, there are traffic lights. These traffic lights are denoted as S_1, S_2, S_3, and S_4. The arrows indicate the directions that should take the vehicles when arriving at the intersection. Each traffic light cycle is composed of four phases as described next:

(1) In the first phase (see Fig. 15), the traffic light S_1 changes to green and the traffic lights S_2, S_3, and S_4 to red, so that the vehicles of segment A can cross the intersection in direction to G, or turn right toward H, or turn left in direction to F.

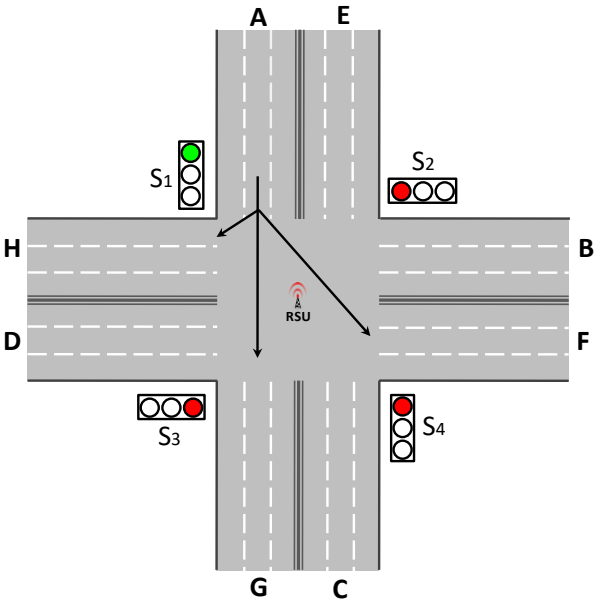

Fig. 15 First Phase of the Traffic Light Cycle at the Intersection

(2) In the second phase (see Fig. 16), the traffic light S_2 changes to green and the traffic lights S_1, S_3, and S_4 to red, so that the vehicles of segment B can cross the intersection in direction to H, or turn right toward E, or turn left in direction to G.

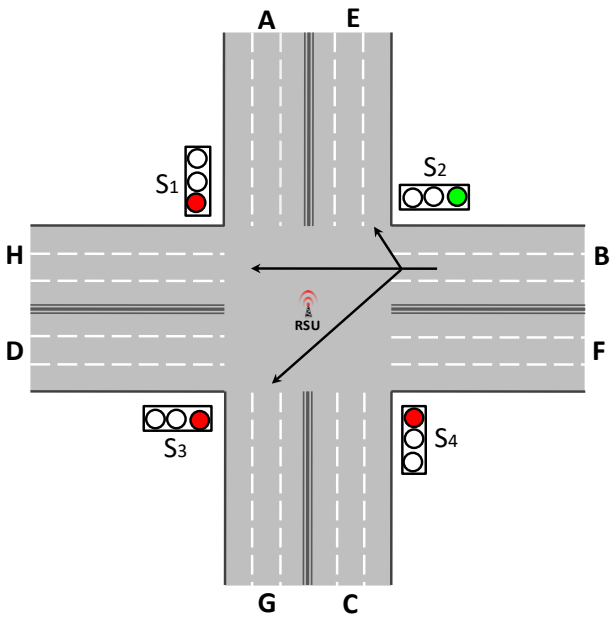

Fig. 16 Second Phase of the Traffic Light Cycle at the Intersection

(3) In the third phase (see Fig. 17), the traffic light S_3 changes to green and the traffic lights S_1, S_2, and S_4 to red, so that the vehicles on segment D can cross the intersection in direction to F, or turn right toward G, or turn left in direction to E.

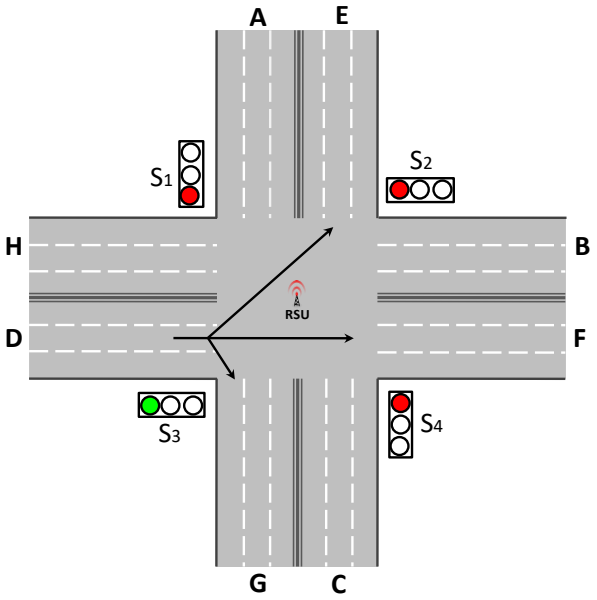

Fig. 17 Third Phase of the Traffic Light Cycle at the Intersection

(4) Finally, in the fourth phase (see Fig. 18), the traffic light S_4 changes to green and the traffic lights S_1, S_2, and S_3 to red, so that the vehicles of segment C can cross the intersection in direction to E, or turn right toward F, or turn left in direction to H.

Additionally, we placed the RSU in the center of the intersection (see Fig. 14). It is worth mentioning that in any of the phases described above (see Figs. 15, 16, 17, and 18), in those segments where the traffic lights change to red (segments B, C, and D in the first phase; segments A, C, and D in the

second phase; segments A, B, and C in the third phase; segments A, B, and D in the fourth phase), the vehicles will stop and, consequently, vehicle queues will be formed and gradually increased in size with the arrival of more vehicles. Now, our algorithm can count vehicles in several directions in parallel. For that, the field *Message Direction* (see Fig. 1) must be set in the COUNT_REQUEST messages. In these experiments, we counted the number of vehicles in segments B, C, and D, respectively, at the same time, during the first phase of the traffic light cycle. For that, the RSU sends three successive COUNT_REQUEST messages. The first one is sent toward the end of segment B (with a *Message Direction* field set to EAST), the second one is sent toward the end of segment C (with a *Message Direction* field set to SOUTH), and the third one is sent toward the end of segment D (with a *Message Direction* field set to WEST). At the beginning of the simulations, the vehicles are distributed in the different lanes of the roads that form the intersection with a total number of units varying from 50 to 500. The RSU starts the counting process with *Hop Limit*=3

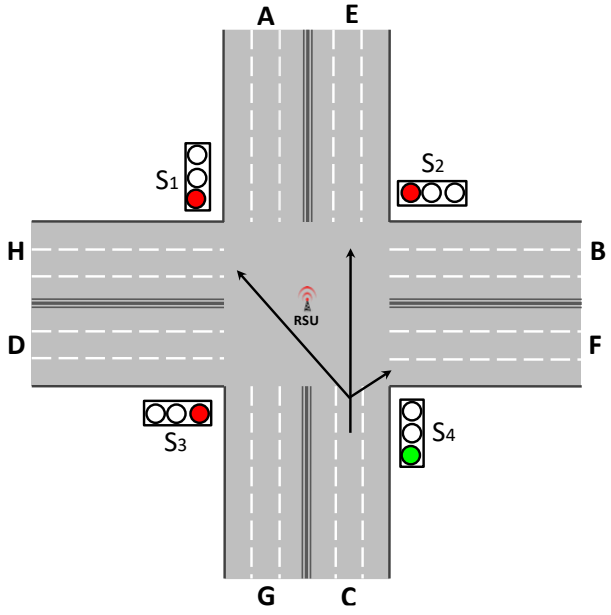

Fig. 18 Fourth Phase of the Traffic Light Cycle at the Intersection

In Tables 5, 6, and 7, we reported results relevant for experiments of the proposed algorithm associated with the number of units counted, the response time, and the total number of control messages sent by the units during the counting, respectively, in scenarios where we varied the number of vehicles (50, 100, 150, 200, 250, 300, 350, 400, 450, and 500) randomly distributed in the three segments (B, C, and D). Additionally, we varied their propagation range: 200m, 300m, and 400m. As already stated, the simulations were done for the first phase of the traffic light cycle (Fig. 15), that is, when the traffic lights S_2 (segment B), S_3, (segment D), and S_4 (segment C) are red, and the traffic light S_1 (segment A) is green. Since the vehicles of segments B, C, and D will stop and wait for the green light, queues will be created in these segments, and increase in size as the time passes with the arrival of new vehicles, while vehicles in segment A will continue their way to segments F, G, or H (see Fig. 19). The results shown in columns B, C, and D of Table 5 are the number of vehicles that should be counted/the number of vehicles actually counted by our algorithm. Table 6 represents the response time in milliseconds (ms), while Table 7 reports the total number of control messages sent by the units during the counting. We can observe from the results of our experiments that our algorithm effectively performs the counting of vehicles in segments B, C, and D, with an adequate response time (see Table 6), and with a low number of control messages sent by the units (see Table 7). These counting results can be used to select the next light phase of the traffic lights at the intersection.

For example, for a total number of vehicles equal to 450 (which were distributed in segments B, C, and D with 144, 142, and 164 vehicles, respectively), and a propagation range equal to 300m, we can see that our algorithm made a counting with a high degree of accuracy in each of the segments (B, C, and D). That is, in segment B, C, and D, the RSU should count 144, 142, and 164 vehicles, respectively, and our algorithm reported 144 (an exact counting), 142 (an exact counting), and 163 (with a margin of error of 0.6%), respectively. According to Table 6, the response times were 9.07ms (segment B), 9.02ms (segment C), and 11.96ms (segment D), and a total number of control messages sent by the vehicles equal to 4, in each of these segments (see Table 7).

It is important to mention that in real vehicular contexts, in fact, vehicle counting can be very helpful during critical periods of flow at an intersection, to make a wiser decision about the light change [34]. Currently, vehicles have to wait a fixed amount of time to get a green signal, even if the other roads at the intersection have no traffic or a light traffic load. This situation can be avoided by programming the lights according to the vehicular density. In other words, the green light should be extended to a longer period for the road where the vehicular density is higher.

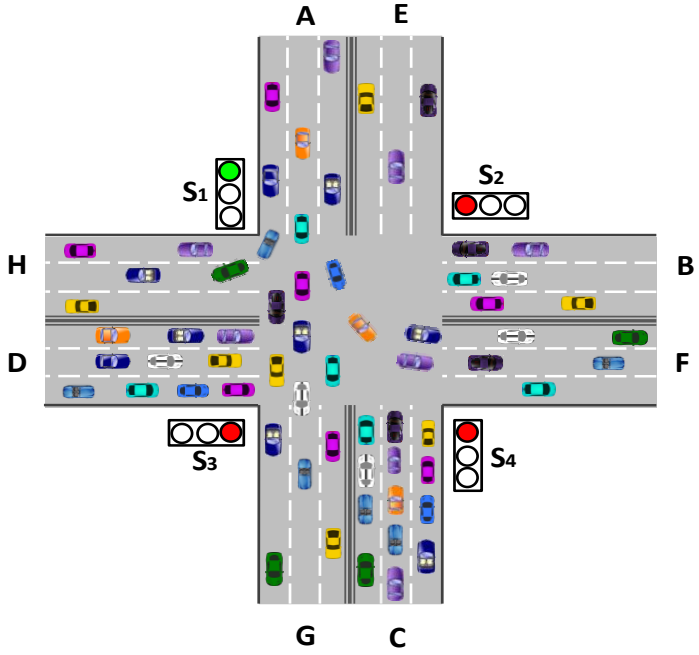

Fig. 19 Example of a Scenario in a Four-way Intersection

Table 5 Units Counted at an Intersection when Varying the Number of Units and Propagation Range

Total Number of Units	Propagation Range Values in Meters								
	200m			300m			400m		
	B	C	D	B	C	D	B	C	D
50	20/20	11/11	19/19	20/20	14/14	16/16	16/16	16/16	18/18
100	29/29	39/39	32/32	30/30	36/36	34/34	30/30	31/31	39/39
150	40/39	60/60	50/49	54/54	48/48	48/48	62/62	38/38	50/50
200	64/63	74/73	62/62	71/71	70/70	59/59	64/64	66/66	70/70
250	87/86	84/82	79/78	91/91	88/87	71/71	77/77	97/97	76/76
300	91/89	99/97	110/108	110/110	103/103	87/86	94/94	110/108	96/96
350	117/116	125/123	108/107	117/117	137/137	95/95	117/116	113/113	120/120
400	142/140	127/127	131/130	137/136	137/137	126/126	154/154	131/131	115/115
450	160/158	144/142	146/145	144/144	142/142	164/163	141/141	142/142	167/166
500	177/175	158/157	165/163	164/163	178/177	158/157	163/163	157/157	180/179

Table 6 Response Time during the Counting at an Intersection when Varying the Number of Units and Propagation Range

Total Number of Units	Propagation Range Values in Meters								
	200m			300m			400m		
	B	C	D	B	C	D	B	C	D
50	3.99ms	3.59ms	3.83ms	3.97ms	3.99ms	4.01ms	3.96ms	3.95ms	3.99ms
100	4.30ms	4.85ms	4.43ms	4.54ms	4.67ms	4.60ms	4.10ms	4.35ms	4.64ms
150	5.12ms	5.80ms	5.23ms	4.33ms	4.25ms	4.23ms	5.68ms	5.27ms	5.52ms
200	10.74ms	10.97ms	7.51ms	7.52ms	13.76ms	7.18ms	5.38ms	5.75ms	5.84ms
250	11.95ms	11.72ms	8.51ms	13.98ms	13.49ms	4.60ms	5.47ms	9.15ms	5.35ms
300	10.78ms	11.50ms	13.49ms	11.35ms	10.26ms	9.05ms	7.25ms	10.32ms	8.58ms
350	16.69ms	16.88ms	12.27ms	7.67ms	7.43ms	6.81ms	5.45ms	8.99ms	9.23ms
400	20.15ms	19.22ms	19.54ms	8.06ms	13.69/0ms	8.19ms	9.45ms	8.99ms	6.23ms
450	22.45ms	22.38ms	21.93ms	9.07ms	9.02ms	11.96ms	9.05ms	9.08ms	10.21ms
500	16.95ms	16.48ms	16.56ms	9.58ms	17.18ms	13.64ms	12.64ms	12.28ms	12.74ms

Table 7 Total Number of Control Messages Sent by the Units during the Counting at an Intersection when Varying the Number of Units and Propagation Range

Total Number of Units	Propagation Range Values in Meters								
	200m			300m			400m		
	B	C	D	B	C	D	B	C	D
50	2	2	2	2	2	2	2	2	2
100	2	2	2	2	2	2	2	2	2
150	2	2	2	2	2	2	2	2	2
200	4	4	3	2	5	2	2	2	2
250	4	4	3	4	4	2	2	3	2
300	4	4	5	4	4	4	3	4	3
350	6	6	4	4	4	2	2	3	4
400	6	6	6	4	5	4	3	3	2
450	6	6	6	4	4	4	4	4	4
500	6	6	6	4	6	5	5	5	5

5.5 Scenarios with Different Penetration Rates

In this section, we study the influence of the penetration rate over the algorithm. To this end, we use the same scenario as the one of Section 5.1 (one-way road of a single lane).

Table 8 shows the counting error as a percentage when we varied the number of vehicles (25, 50, 75, 100, 125, 150, and 175) and the penetration rate (100%, 95%, 90%, 85%, and 80%), with a propagation range of 300m. The simulations seem to indicate that the counting error is proportionally affected by the decrease of the penetration rate, i.e., for a penetration rate of 80%, the counting error fluctuates around 20% (which is 100%-80%). These results are encouraging, since the algorithm still makes an effective counting of the units that do have an RSU, even in the presence of not

VANET-based vehicles. It is worth to mention that there is no way to count a vehicle that does not have an RSU when we limit the system to the WAVE and GPS technologies. Aggregating some other information from other sensors of the vehicles or from a few "in-situ" sensors on the roadside might reduce the error counting, but this possible research is outside the scope of this paper.

Table 8 Counting Error in Percent when Varying the Number of Units and Penetration Rate (Road with One Lane)

Total Number of Units	Penetration Rates				
	100%	**95%**	**90%**	**85%**	**80%**
25	0.0	4.9	10.0	15.1	19.8
50	0.0	5.0	9.7	14.8	20.1
75	0.2	5.2	10.3	15.2	20.0
100	0.5	4.7	9.2	15.3	20.2
125	1.1	5.5	9.1	14.3	19.1
150	1.7	6.1	11.8	16.9	21.0
175	1.8	6.9	12.3	17.2	22.3

6 Conclusions and Future Work

The major contribution that we have achieved in this paper is the design and implementation of an efficient novel algorithm to count vehicles that are stopped at a traffic light, by using VANET technology. This algorithm can be used as a basic tool in the development of Adaptive Traffic Control Systems (ATCSs), and should dramatically help to optimize vehicular flow.

The proposed algorithm was simulated in different scenarios using SUMO and OMNeT++ as simulators, and Veins as a framework to bi-directionally couple the simulators. To evaluate its performance, we conducted two different sets of experiments. In the first set of experiments, we evaluated the performance of our algorithm in scenarios where we varied the total number of units and their respective propagation range in one-way roads of one, two, and three lanes. In the second set of experiments, we evaluated the behavior of the algorithm in a four-way intersection, with several lanes.

The simulations that we performed show that our algorithm efficiently calculates a total number of vehicles, with very low response times and small numbers of control messages (COUNT_REQUEST and COUNT_REPLY) sent by the units during the counting.

As possible future work, we plan to enhance our algorithm by dividing roads into "segments" or "regions of counting" of fixed or variable size, where a segment leader will be designated and be in charge of counting the vehicles in its respective segment, with the purpose of minimizing the response time. We also intend to implement our algorithm under a radio propagation model with random behavior and variations in the link qualities from one transmission to the next, in order to study and analyze its influence on the results. In the same direction, we project to study the impact of a 10- to 15-meter error over the positions reported by GPSs, in the algorithms. Finally, we are also interested in the development of a complete procedure for more intelligent signal timing strategies to improve traffic capacity at intersections [35], based on the counting algorithm presented in this paper.

Acknowledgments

We thank the CDCH-UCV (Consejo de Desarrollo Científico y Humanístico) which partially supported this research under grant number: PG 03-8066-2011/1.

References

[1] E. Gamess and I. Mahgoub. A Novel VANET-Based Approach to Determine the Position of the Last Vehicle Waiting at a Traffic Light. In *Proceedings of the 2011 International Conference on Wireless Networks (ICWN'11)*, Las Vegas, Nevada, USA, 2011, pp. 327-333.

[2] S. Najafzadeh, N. Ithnin, S. Abd Razak, and R. Karimi. Dynamic Broadcasting in Vehicular Ad hoc Networks. *International Journal of Computer Theory and Engineering*, Vol. 5, No. 4, pp. 629-632, 2013.

[3] R. Soto. Redes Vehiculares AdHoc – VANET. Boletín CIIAS (Centro de Integración para la Industria Automotriz y Aeronáutica de Sonora, A.C.). No. 047, Abril 2009.

[4] D. Jiang and L. Delgrossi, "IEEE 802.11p: Towards an International Standard for Wireless Access in Vehicular Environments," in *Proceedings of the 2008 IEEE 67th Vehicular Technology Conference (VTC Spring 2008)*. Marina Bay, Singapore, May 2008, pp. 2036–2040.

[5] Y. Li, "An Overview of the DSRC/WAVE Technology," in Quality, Reliability, Security and Robustness in Heterogeneous Networks, Vol. 74 of Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Springer Berlin Heidelberg, 2012, pp. 544–558.

[6] R. Uzcategui and G. Acosta-Marum, "WAVE: A Tutorial," *IEEE Communications Magazine*, Vol. 47, May 2009, pp. 126–133.

[7] S. Yi, Design and Construction of LAN based Car Traffic Control System, *World Academy of Science, Engineering and Technology*, Issue 46, pp. 612-615, October 2008.

[8] S. Dornbush and A. Joshi, StreetSmart Traffic: Discovering and Disseminating Automobile Congestion using VANETs. In *Proceedings of the 2007 IEEE 65th Vehicular Technology Conference (VTC Spring 2007)*, Dublin, Ireland, April 2007.

[9] A. Ghazy and T. Ozkul, Design and Simulation of an Artificially Intelligent VANET for Solving Traffic Congestion. In *Proceedings of the International Symposium on Mechatronics and its Applications (ISMA'09)*, Sharjah, United Arab Emirates, March 2009.

[10] F. Padron and I. Mahgoub, Traffic Congestion Detection Using VANET. Florida Atlantic University, Tech. Rep. 2010.

[11] E. Kell and E. Mills, Traffic Detector Handbook. U.S. Department of Transportation, Federal Highway Administration, 2nd Edition, pp. 1-39. USA, 1990.

[12] L. Klein, Sensors Technologies and Data Requirements for ITS Applications, Artech House Publishers, Norwood, USA, June 2001.

[13] L. Mimbela and L. Klein, A Summary of Vehicle Detection and Surveillance Technologies Used in Intelligent Transportation Systems. Handbook, Federal Highway Administration, Intelligent Transportation Systems, USA, 2007.

[14] S. Grafling, P. Mahonen, and J. Riihijarvi, Performance Evaluation of IEEE 1609 WAVE and IEEE 802.11p for

Vehicular Communications. In *Proceedings of the 2010 Second International Conference on Ubiquitous and Future Networks (ICUFN 2010)*, pp. 344–348, Jeju Island, Korea, June 2010.

[15] G. Leduc, Road Traffic Data: Collection Methods and Applications. European Commission, Joint Research Center, Institute for Prospective Technological Studies, Seville, Spain, 2008.

[16] N. Chintalacheruvu and V. Muthukumar, Video Based Vehicle Detection and Its Application in Intelligent Transportation Systems, *Journal of Transportation Technologies*, Vol. 2, No. 4, pp. 305-314, September 2012.

[17] C. Harris and M. Stephens, A Combined Corner and Edge Detector. In *Proceedings of the 4th Alvey Vision Conference (AVC'88)*, Manchester, United Kingdom, September 1988.

[18] M. Lei, D. Lefloch, P. Gouton, and K. Mfadani, A Video-Based Real-Time Vehicle Counting System using Adaptive Background Method. In *Proceedings of the 4th IEEE International Conference on Signal Image Technology and Internet Based Systems*, Bali, Indonesia, November 2008.

[19] M. Tursun and G. Amrulla, A Video Based Real-Time Vehicle Counting System using Optimized Virtual Loop Method. In *Proceedings of the 2013 International Workshop on Systems, Signal Processing and their Applications (WoSSPA)*, Algiers, Algeria, May 2013.

[20] K. Peiris and D. Sonnadara, Extracting Traffic Parameters at Intersections through Computer Vision. In *Proceedings of the Technical Sessions*, Vol. 27, pp. 68–75, 2011.

[21] A. Knaian, "A Wireless Sensor Network for Smart Roadbeds and Intelligent Transportation Systems," Master Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, June 2000.

[22] M. Litzenberger, B. Kohn, G. Gritsch, N. Donath, C. Posch, N.A. Belbachir, and H. Garn, Vehicle Counting with an Embedded Traffic Data System using an Optical Transient Sensor. In *Proceedings of the 10th International IEEE Conference on Intelligent Transportation Systems (ITSC'07)*, Seattle, Washington, USA. September 2007.

[23] M. Contreras and E. Gamess. A Multi-Interface Multi-Channel Algorithm to Count Nodes Using Wireless Technology. American Journal of Networks and Communications, Vol. 6, No. 1, pp. 1-19, February 2017.

[24] Q. Luo, S. Wei, H. Cheng, and M. Ren, A Cooperative Framework for Region Crowdedness Sensing in VANETs, In *Proceedings of the 2017 IEEE/CIC International Conference on Communications in China (ICCC 2017)*, Qingdao, China, October 2017.

[25] IEEE 1609 – Family of Standards for Wireless Access in Vehicular Environments (WAVE), U.S. Department of Transportation, January 2006.

[26] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins, Global Positioning System: Theory and Practice, 5th Edition, Springer, September 2004.

[27] F. Martinez, C. Toh, J. Cano, C. Calafate, and P. Manzoni, A Survey and Comparative Study of Simulators for Vehicular Ad Hoc Networks (VANETs), *Wireless Communications and Mobile Computing*, Vol. 11, No. 7, pp. 813–828. July 2011.

[28] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, SUMO - Simulation of Urban MObility: An Overview. In *Proceedings of the Third International Conference on Advances in System Simulation (SIMUL 2011)*, Barcelona, Spain, October 2011.

[29] A. Varga and R. Hornig, An Overview of the OMNeT++ Simulation Environment. In *Proceedings of the First International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools 2008)*, Marseille, France, March 2008.

[30] E. Gamess and M. Contreras, "A Proposal for an Algorithm to Count Nodes using Wireless Technologies". *International Journal of High Performance Computing and Networking*, Vol. 8, No. 4, 2015, pp. 345-357.

[31] C. Sommer, R. German, and F. Dressler, Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis, *IEEE Transactions on Mobile Computing*, Vol. 10, No. 1, pp. 3–15. January 2011.

[32] A. Wegener, M. Pi Orkowski, M. Raya, H. Hellbruck, S. Fischer, and J.-P. Hubaux, TraCI: An Interface for Coupling Road Traffic and Network Simulators. In *Proceedings of the 11th Communications and Networking Simulation Symposium (CNS 2008)*, Ottawa, ON, Canada, April 2008.

[33] K. Wessel, M. Swigulski, A. Kopke, and D. Willkomm, MiXiM: The Physical Layer An Architecture Overview. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques (SIMUTools 2009)*, Rome, Italy. March 2009.

[34] W. S. Homburger, J. W. Hall, R. C. Loutzenheiser, and W. R. Reilly, Volume Studies and Characteristics. In Fundamentals of Traffic Engineering. UC Berkeley Institute of Transportation Studies, University of California at Berkeley, pp. 5.1–5.6. 1996.

[35] X. Huang, Q. Zhang, and Y. Wang, Research on Multi-agent Traffic Signal Control System based on VANET Information, in *Proceedings of the 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC 2017)*, Yokohama, Japan, October 2017.