American Journal of Networks and Communications
2017; 6(1): 1-19
http://www.sciencepublishinggroup.com/j/ajnc
doi: 10.11648/j.ajnc.20170601.11
ISSN: 2326-893X (Print); ISSN: 2326-8964 (Online)

A Multi-interface Multi-channel Algorithm to Count Nodes Using Wireless Technology

Manuel Contreras[1], Eric Gamess[1, 2]

[1]School of Computer Science, Central University of Venezuela, Los Chaguaramos, Caracas, Venezuela
[2]Department of Computer Science, University of Puerto Rico, Rio Piedras, Puerto Rico

Email address:
mcontre@ula.ve (M. Contreras), eric.gamess@upr.edu (E. Gamess)

To cite this article:
Manuel Contreras, Eric Gamess. A Multi-interface Multi-channel Algorithm to Count Nodes Using Wireless Technology. *American Journal of Networks and Communications.* Vol. 6, No. 1, 2017, pp. 1-19. doi: 10.11648/j.ajnc.20170601.11

Received: September 28, 2016; **Accepted**: October 10, 2016; **Published**: February 23, 2017

Abstract: In wireless networks, devices can be equipped with multiple interfaces to utilize multiple channels and increase the aggregated network throughput. In fact, as the current price of network interface cards has fallen dramatically, applications have started to use multiple non-overlapping channels to get an enhanced bandwidth, with traditional standards such as IEEE 802.11 a/b/g. In this regard, a wireless network node equipped with more than one interface can concurrently communicate with other nodes on different channels. This operation results in less interference and collisions in the network, and therefore a better use of the network capabilities in terms of bandwidth. In this paper we propose an algorithm that uses multiple channels to improve performance in the counting of objects (people, animals, devices, vehicles, etc) based on wireless communications where devices are equipped with multiple interfaces, which works either for stationary nodes or in scenarios where nodes are moving even at high speeds. In particular, the technique of interface switching is used to take advantage of all the channels, even when the number of available interfaces is smaller than the number of channels. To validate and evaluate the performance and accuracy of the proposal, the algorithm is simulated using a famous network simulation tool called OMNeT++/INET. The results of the simulations show that the proposed algorithm efficiently exploits the advantages of multi-channel, by computing a number of nodes very close to the real one (even in the case of scenarios with nodes moving at high speeds) with an acceptable response time and total number of control messages sent by the nodes to accomplish the counting task.

Keywords: Wireless Networks, Multi-channel Networks, Multi-radio Networks, Network Interface Cards, Node Counting, OMNeT++, INET, Network Simulator

1. Introduction

As an emerging and promising technology, wireless networks have a wide range of potential applications. These networks typically make use of a single radio interface on a fixed channel to communicate with neighboring nodes. However, previous research has advocated the usage of multiple wireless channels to increase the aggregated network throughput and provide reliable and timely communication services. Wireless spectrum is divided into multiple channels by industry standards for two main reasons: (1) to allow parallel utilization of the spectrum by multiple wireless technologies at the same time, and (2) the design of wideband wireless transceivers is very complex because of the frequency dependent components involved in the design [1]. IEEE 802.11 is a widely used technology for wireless local area networks and current Network Interface Cards (NICs) are capable of communicating over multiple non-overlapping channels. In existing system architectures, the use of multiple channels can be supported by providing every channel with a dedicated interface. Multiple non-overlapping channels (also known as orthogonal channels) exist in the 2.4 GHz and 5 GHz spectrum. For example, IEEE 802.11b has 11 channels in the 2.4 GHz spectrum, 3 of which are orthogonal, while IEEE 802.11a has 12 orthogonal channels in the 5 GHz spectrum, and IEEE 802.11g standard defines 3 orthogonal channels in the 2.4 GHz spectrum. Traditionally, these channels are used by different networks operating in the same vicinity. However, it is possible to concurrently take advantage of these channels by using multiple transceivers or radios per device [1]. Effective

utilization of these multiple channels would increase the bandwidth substantially. Such wireless networks with Multi-Interface Multi-Channel (MIMC) devices are known as MIMC wireless networks. They are used to build mesh networks [2], vehicular ad hoc networks [3], among others, so that a new possibility has been opened to develop algorithms for counting nodes based on these emerging technologies. However, most IEEE 802.11-based multi-hop ad hoc networks today use only a single channel. As a result, these networks can rarely fully exploit the aggregate bandwidth available in the radio spectrum provisioned by the standards [4]. Hence, wireless networks that use multiple radios in a collaborative manner dramatically improve system performance and functionality over the traditional single radio wireless systems that are popular today [5].

In this research work, we introduce an algorithm to count nodes using wireless communications where nodes are equipped with multiple interfaces working on multiple channels. For this primary version of the algorithm, it is assumed that 2 NICs are available at each node. One interface has a fixed channel to receive control messages from neighbors and the other interface will switch among the channels to transmit control messages to neighbors. This algorithm can be used as a basic and integral tool for the development of applications in many fields. For example, in the field of ATCSs (Adaptive Traffic Control Systems), if the number of vehicles present in each lane at the intersections of roads is known in real time, the cycles of their traffic lights can be continuously optimized or adjusted, and thus achieving a greater vehicular flow with balanced and fair waiting times. Another application of the algorithm could be the counting of people attending an event (musical, political, or social), in public spaces, stadiums, etc., for security and billing purposes, that would be based on the counting of the number of cell phones with a certain network technology such as WiFi. Another relevant use of the algorithm lies in the context of parking management systems with the intention to count the total number of vehicles currently parked so the number of available slots can be computed, allowing a better control for admission of vehicles at the entrance of the parking.

With the aim of validating the algorithm, a discrete event simulator called OMNeT++/INET is used to test and analyze our proposal in different scenarios, where we varied different parameters such as the speed of nodes, the density of nodes, the signal propagation range, etc. The results of the simulations show that the algorithm performs an effective counting of nodes with a quite acceptable response time and number of control messages sent by the nodes, in both stationary scenarios and in scenarios where nodes are moving at high speeds.

The rest of this paper is organized as follows. Section 2 reviews the related work. In Section 3, we introduce our multi-interface multi-channel algorithm to count nodes based on wireless technologies. Section 4 briefly describes the simulation tools and scenarios that we use to study and validate the proposed algorithm. Section 5 presents an analysis of the performance results of our simulations. Finally, Section 6 concludes the paper and presents future work.

2. Background and Related Work

A lot of recent research and commercial efforts in the wireless networking industry have focused on a new class of ad hoc networks: multi-interface multi-channel ad hoc networks [5]. These networks are characterized by a set of nodes with multiple wireless interfaces utilizing multiple orthogonal channels at the same time over a given swath of spectrum such as IEEE 802.11 a/b/g.

At present, in the specialized literature there are many related efforts that study the benefit of using multiple channels. Even though multiple non-overlapping channels exist in the 2.4 GHz and 5 GHz spectrum, most IEEE 802.11-based multi-hop networks nowadays use only a single channel, and therefore only require a single interface. A well-known fact that is affecting the performance of such networks is the significant throughput degradation along the multi-hop path [5]. When a single channel is used for both incoming and outgoing traffic, throughput is halved as they use the same radio channel. This means that when one node is transmitting, its neighbor nodes must all be in listening mode otherwise a collision will occur. This problem is amplified across the network, and after a few hops the bandwidth is reduced significantly. To overcome the throughput degradation problem, a natural approach is to use multiple channels simultaneously in order to reduce collisions [5].

Multiple channels can be exploited by using a single radio (or interface) per device or by having multiple radios per device. In the former scenario, two devices wishing to communicate tune their radios to the same channel and exchange information while other devices, in the vicinity, would be tuned to other channels. In the latter scenario, two devices can potentially tune to multiple channels at the same time, using multiple radios, and communicate on multiple channels simultaneously. The decision to use a single radio or multiple radios per device depends on the implementation requirements, dictated by various factors including ease of deployment, bandwidth requirement, and cost [1].

Moreover, the multi-hopping technique increases the network range and scalability but it is associated with a high level of interferences. In fact, capacity reduction is a well-known issue derived from the interference problem in multi-hop wireless networks. This problem can be mitigated by using the multi-interface multi-channel technique [6]. As the current price of NICs has quickly dropped, the existing standards, such as IEEE 802.11 a/b/g, opt for multiple non-overlapping channels. In this regard, a node equipped with more than one interface can concurrently communicate with other nodes on different channels. This operation results in less interference in the network [7].

Raniwala and Tzi-cker [4] have experimentally shown that when separation between interfaces is increased, the

interference between interfaces is reduced, allowing more channels to be used simultaneously. So and Vaidya [8] proposed a scheme that allows wireless devices to communicate on multiple channels using a single interface card. The scheme requires frequent channel switching, which implies considerable overhead on the current hardware. The authors in [9] [10] [11] presented a multi-channel carrier sense multiple access MAC protocol where all nodes have an interface on each channel. The proposed protocol uses different metrics to choose the channel for communication between nodes. Wu et al. [12] developed a protocol that assigns channels dynamically, in an on-demand style that requires two interfaces. One interface is assigned to a common channel for control purposes, and the second interface is switched between the remaining available channels for transmitting data. Lee, Midkiff, and Park [13] introduced a proactive routing protocol on multiple channels, which uses one control channel and *n* data channels. The nodes exchange control packages on the control channel to negotiate the best channel for receiving in real time. In [14] [15], a hybrid channel assignment scheme is proposed where some radios are statically assigned a channel while the remaining radios can dynamically change their frequency. The authors also presented a new routing strategy based on channel switching and route diversity cost. As an example, HMCP (Hybrid Multi-Channel Protocol) is a hybrid method which exploits both static and dynamic channels [16]. Recent studies [17] [18] show that hybrid approaches exploit the advantages of both static and dynamic methods.

On the other hand, the detection and counting of nodes has been of great interest in various fields and can be used as a basic tool in the development of many applications such as: (1) the counting of people in public spaces, stadiums, musical, political, or social events for security and billing purposes, (2) the estimation of the population of animal species that are in danger of becoming extinct for their protection, and (3) the determination of the number of free parking slots in a parking lot to allow or not the entrance of additional vehicles. At the present time, most of the alternatives or solutions to detect and count nodes, with lesser or greater accuracy, are based on methods and techniques supported by conventional "in-situ" technologies (turnstiles, barrier arms, digital cameras, video cameras, thermal cameras, pneumatic road tubes, magnetic sensors, infrared beams, etc) [19], as reported in the specialized literature. Chao-Ho et al. [20] proposed a method for counting people entering or leaving a bus based on video processing, where a zenithal camera is set in the bus for capturing the bi-directionally passenger flow. In [21] the author presented a system for counting people in movement passing through a given area (a concourse, a tunnel, or a gate) with an infrared sensor network. The system does not only report the number of people passing through, but also provides the corresponding moving direction of every person. Kim et al. [22] introduced a people counting system with a single fixed camera which detects and tracks moving people. This system counts the number of moving objects (people) entering a security door. Marques et al. [23] provided a framework for passive acoustic-based density estimation, illustrated with examples from real-world case studies. They focused on methods involving sensors at fixed locations, particularly hydrophones, to estimate population density or abundance, based on detecting sounds naturally produced by animals. The authors of [24] presented two algorithms that count birds with wireless sensors equipped with microphones that listen for bird songs in the intersection of their detection areas, and compute an approximation of the number of singing birds for the monitored period. Similarly, in the field of vehicular transportation, Knaian [25] developed a wireless sensor package to monitor roadways in the Intelligent Transportation Systems (ITSs) to count passing vehicles, measure the average roadway speed, and detect ice and water on the road.

To the best of our knowledge, in the field of counting objects based on wireless communications, just a few works has been developed. For example, Gamess and Contreras [26] introduced an algorithm to count nodes or devices equipped of a wireless network interface, using wireless technologies. In [27], the authors presented an improved version of their algorithm proposed in [26], where they enhanced the accuracy of the counting by taking into account late counting messages. The authors in [28] proposed a novel VANET-based approach to obtain: (1) the position of the last vehicle and (2) the number of vehicles, in a line of vehicles stopped at a traffic light. To compute the number of nodes, the authors first obtain the length of the queue of vehicles stopped at the traffic light and then divide this distance by a constant value (7 meters).

According to our study, we can see that most of the research done in the field for counting objects is primarily based on the usage of "in-situ" technologies. In this work, we propose a new algorithm to count nodes equipped of 2 wireless NICs, that substantially differ from [26][27] by taking advantage of the multi-interface multi-channel wireless technologies, with the aim of reducing the number of collisions in the network.

3. Algorithm to Count Nodes Using Multi-interface Multi-channel Wireless Communications

In this section, we describe a multi-interface multi-channel algorithm to count nodes using wireless networks. We suppose that each node is equipped with two communication interfaces. The first interface is utilized to receive incoming control messages. Each node uses its own fixed channel in this interface (called *fixedChannelNode*), that can be different from the one of another node. The goal of the second interface is the sending of outgoing control messages to other nodes. The channel of this second interface will vary according to the fixed channel of the first interface of the intending receiver node. Also, in this version of the algorithm, it is assumed that the mobile nodes never turn off

their NICs for energy saving. It is not unrealistic, since some networks always maintain on the NICs, such as the vehicular networks, with the aim to guarantee the physical security of the drivers and passengers.

3.1. Basic Considerations for Counting Nodes in a Multi-interface Multi-channel Wireless Network

The algorithm described in this work is loosely based on the ideas presented in the work developed in [26][27]. As we explain later, we have introduced a neighbor discovery protocol and significantly improved the propagation of COUNT_REPLY messages back to the originator.

3.1.1. Neighbors Discovery Protocol

The protocol proposed in this paper is designed for a multi-hop wireless network. Nodes in the network can be mobile. To track 1-hop neighbors, a neighbor discovery protocol was added to the nodes. Each node periodically broadcasts a BEACON message (see Fig. 1) that contains the sending timestamp, its actual position and speed (obtained from its GPS receiver so that 1-hop neighbors are aware of its presence and location), its *nodeToGoBack*, and *fixedChannelNode* (fixed channel being used by this neighbor). The synchronization of time between the different nodes is solved with the time received from the GPS satellites. *nodeToGoBack* stores a reference of the node to go back toward the originator for the sender of the BEACON message. *fixedChannelNode* is a reference of the fixed channel used by the first interface of the sending node. Based on the received BEACON messages, a node establishes a list of 1-hop neighbors (*NeighborTable*). For each 1-hop neighbor, the node stores its ID (IPv4 address), timestamp, position, speed, *nodeToGoBack*, and *fixedChannelNode* in table *NeighborTable*. With this information, the node can interpolate the actual position of its 1-hop neighbors at any time. When a node receives a new BEACON message from a 1-hop neighbor, the node updates the associated entry in its *NeighborTable*, so that the last updated information is always present in this table, which is used to update the *ChannelUsageList* list. *ChannelUsageList* is a list of *n* counters, where *n* is the total number of orthogonal channels supported by the wireless technology used (e.g., 3 in IEEE 802.11g for channels 1, 6, and 11). The first counter will count the number of 1-hop neighbors that have chosen the first orthogonal channel for their first interface (fixed channel). The second counter will count the number of 1-hop neighbors that have chosen the second orthogonal channel for their first interface, and so on. If a node does not receive three consecutive BEACON messages from a specific 1-hop neighbor (maybe because it has moved out of range), it removes this 1-hop neighbor from its list of 1-hop neighbors (*NeighborTable*) and also updates *ChannelUsageList* accordingly.

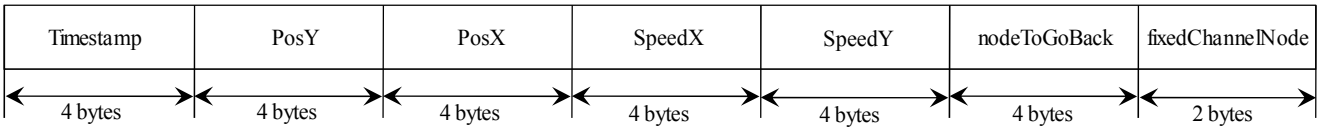

Figure 1. *BEACON message.*

3.1.2. Timing Diagram for the Algorithm

In this paper, we call "*originator*" the node that starts the counting process, i.e., the node that requires the number of nodes around it, up to a specified hop count (called *HopLimit* in the algorithm). Beside of the neighbor discovery protocol described before, the basic approach of the algorithm is:

a. Propagate a broadcast message (called COUNT_REQUEST) from the originator to nodes that are far away from the originator with the number of *HopAway* the receiver of the message is from the originator of the COUNT_REQUEST. To improve or optimize the number of COUNT_REQUEST messages sent by a node, the node will decide to send 1, 2, or 3 COUNT_REQUEST messages depending on the number of COUNT_REQUEST messages that it has received from its neighboring nodes, i.e., if the node determines that it has received a considerable number of COUNT_REQUEST messages, then the node will send a low number of COUNT_REQUEST messages with the aim of saving bandwidth in its neighborhood.

b. Propagate unicast messages (COUNT_REPLY) from the nodes that are far away from the originator toward the originator with the total number of nodes counted up to now (called *Total* in the algorithm).

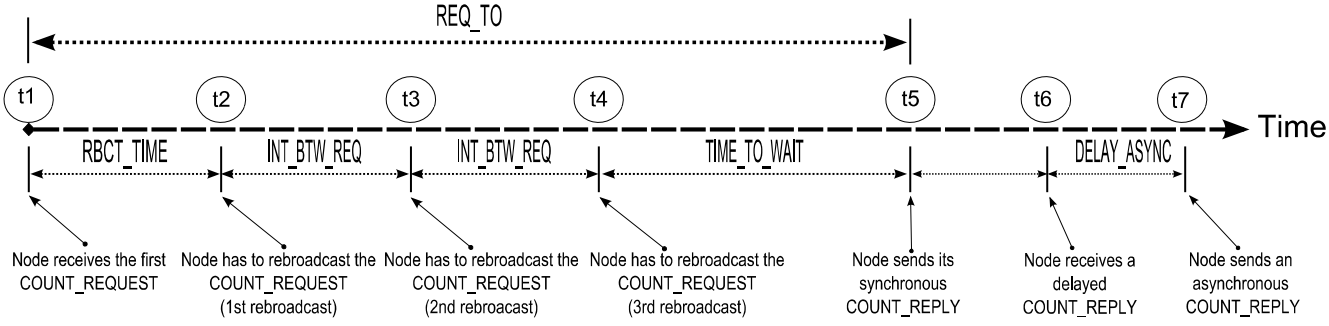

Figure 2. *Timing diagram.*

Fig. 2 depicts a timing diagram related to the propagation of the broadcast messages (COUNT_REQUEST) and the unicast messages (COUNT_REPLY). A description of the times involved (t1, t2, t3, t4, t5, t6, and t7) can be found in Table 1.

Table 1. *Description of times used for the propagation of COUNT_REQUEST and COUNT_REPLY messages.*

t1 = Time when a node receives the first COUNT_REQUEST
t2 = t1 + RBCT_TIME
t3 = t1 + RBCT_TIME + 1*INT_BTW_REQ
t4 = t1 + RBCT_TIME +2*INT_BTW_REQ
t5 = t1 + REQ_TO
t6 = Time when the actual node receives a possible delayed COUNT_REPLY message from another node.
t7 = Time when the actual node sends a possible asynchronous COUNT_REPLY message.

Fig. 2 also shows that the nodes will rebroadcast up to three COUNT_REQUEST messages in row and also send a synchronous COUNT_REPLY message after a specific time. That is, in the worst case it will be necessary to send several times (3 times in the algorithm) the COUNT_REQUEST message, since it is a broadcast message that can collide with other messages without this being detected. In this new algorithm, each node *X* sends its COUNT_REPLY message to the originator directly or to another node (closer to the originator), using the following considerations:

- If the originator is in the list of 1-hop neighbors and if it is still in the propagation range of node *X* (according to the computation of the distance between node *X* and the originator), then node *X* must send its COUNT_REPLY message directly to the originator.
- Otherwise, if there is a node (let call it as node *A*) in the list of 1-hop neighbors of node *X*, and if node *A* is still in its propagation range (according to the computation of the distance between nodes *X* and *A*), and node *A* has a *nodeToGoBack* equal to the originator, then node *X* must send its COUNT_REPLY message to node *A*.
- Otherwise, if node *A* is the *nodeToGoBack* of node *X*, and node *B* is the *nodeToGoBack* of node *A*, and node *B* is in the list of 1-hop neighbors of node *X*, and node *B* is still in the propagation range of node *X* (according to the computation of the distance between nodes *X* and *B*), then node *X* must send its COUNT_REPLY message directly to node *B*, in order to prevent 1 unnecessary hop (node *X* to node *A* and then, node *A* to node *B*). It is also possible to go beyond and avoid 2 unnecessary hops, when node *C* is the *nodeToGoBack* of node *B*, and node *C* is in the list of the 1-hop neighbors of node *X*, and node *C* is still in the propagation range of node *X* (according to the computation of the distance between nodes *X* and *C*), then node *X* must send its COUNT_REPLY message directly to node *C*. This can be generalized to avoid 3, 4, 5, or more unnecessary hops.
- Otherwise, if *nodeToGoBack* is in the list of 1-hop neighbors of node *X* and if it is still in the propagation range of node *X* (according to the computation of the distance between node *X* and *nodeToGoBack*), then node *X* must send its COUNT_REPLY message to *nodeToGoBack*.
- Otherwise, when node *X* cannot send its COUNT_REPLY message to its *nodeToGoBack* because it is out of its propagation range, node *X* has to select a new destination node in the path toward the originator. From its list of 1-hop neighbors, node *X* chooses its new destination as the node that is currently closest in distance to the originator, i.e. by calculating the distance between the actual position of the 1-hop neighbors and the actual position of the originator, using the information (*Timestamp*, *PosX*, *PosY*, *SpeedX*, *SpeedY*) gotten in the BEACON messages (see Fig. 1), and the information of the originator (*OrgTS*, *OrgX*, *OrgY*, *OrgSX*, *OrgSY*) received in the COUNT_REQUEST messages (see Fig. 4). This distance is computed applying the following formula:

$$distanceOrigin = \sqrt{(newOrgX - newPosX)^2 + (newOrgY - newPosY)^2} \quad (1)$$

Where:

newPosX= *PosX*+(*SpeedX**(*ActualTime-Timestamp*))

newPosY= *PosY*+(*SpeedY**(*ActualTime-Timestamp*))

newOrgX= *OrgX*+(*OrgSX**(*ActualTime-OrgTS*))

newOrgY= *OrgY*+(*OrgSY**(*ActualTime-OrgTS*))

- Now, if node X does not have any 1-hop neighbor currently (i.e., its list of 1-hop neighbors is empty), it will plan a new attempt to send its COUNT_REPLY message in the near future.
- An asynchronous COUNT_REPLY message is sent by a node with a slight delay called DELAY_ASYNC, after receiving a delayed COUNT_REPLY message. This approach forces the cooperating nodes to update the actual counting of nodes, reaching a more accurate result [27].
- The other parameters that are also used in the algorithm are described here (see Fig. 2):
- RBCT_TIME (Rebroadcast Time): Time between the reception of the first COUNT_REQUEST, and the first rebroadcast of the COUNT_REQUEST by the actual node.
- INT_BTW_REQ (Interval Between Request): Time interval between the sending of COUNT_REQUEST messages. In other words, it also represents the time between two consecutive COUNT_REQUEST messages sent by the actual node.
- REQ_TO (Request Timeout): It is the time between the reception of the first COUNT_REQUEST and the

moment when the actual node has to send the synchronous COUNT_REPLY message to a node, toward the originator.

- TIME_TO_WAIT: It is the time a node waits after the rebroadcast of the third COUNT_REQUEST message and the sending of the synchronous COUNT_REPLY message to a node, toward the originator. This time must be big enough to allow the propagation of COUNT_REQUEST messages from the actual node toward nodes that are far away from the originator, and the propagation of the COUNT_REPLY messages from the nodes that are far away from the originator toward the actual node. By the way, TIME_TO_WAIT is not a constant value and will be computed by every node according to *HopLimit* and how far away it is from the originator (called *HopAway* in the algorithm).
- DELAY_ASYNC: A small random time that a node waits after the reception of a delayed COUNT_REPLY message, and before the sending of the asynchronous COUNT_REPLY message.

3.1.3. First Interface

A static channel is assigned to the first interface. This channel is also called fixed channel or reception channel. There are two steps involved in the static channel assignment: (1) choosing the channel to be assigned to the first interface, and (2) informing neighbors about the fixed channel being used by the actual node through the transmission of BEACON messages. It ensures that a node intending to communicate with its 1-hop neighbors can switch its second interface to their fixed channel before the transmission. To balance the usage of channels, it is beneficial if other nodes in the neighborhood use different channels for their first interface. For this, each node maintains a table named *NeighborTable* containing the fixed channels being used by its 1-hop neighbors. Nodes also maintain a list named *ChannelUsageList* containing the number of 1-hop neighbors that are listening control messages in each orthogonal channel.

3.1.4. Second Interface

The second interface can be switched dynamically among multiple channels, also called as varying channels or transmission channels. This interface is mainly used for transmission, and the sender must switch the interface to the fixed channel of the receiver before transmission [29].

3.1.5. Fixed Channel Assignment Procedure

The purpose of fixed channel assignment is to specify the receive channel of the current node for control messages, with the goal of minimizing interference with other nodes. In this regard, we apply the hybrid scheme [30] that divides the NICs on a node in two: one fixed NIC and one switchable NIC. While the fixed NIC is assigned to a particular channel, the switchable NIC switches over different orthogonal channels as required. In Fig. 3, we described the channel assignment procedure used by each node in the network to determine the channel used by its fixed interface.

```
1. /* Channel Assignment Procedure for Fixed Interface
      input: NeighborTable of the node                    */
2. for each Neighbor in NeighborTable do
3.    if Neighbor in NeighborTable uses Channel[j] then
4.       ChannelUsageList[j]= ChannelUsageList[j] + 1;
5.    end if
6. end for
7. Find the channel with minimum counter in ChannelUsageList and
   assign it to fixedChannelNode (fixed interface) of the node
```

Figure 3. *Fixed channel assignment procedure.*

The assignment of the fixed channel to the first interface of a particular node takes place with the following steps:

a. Initially, each node is assigned with a random channel in its fixed interface and then the fixed channel allocation procedure (see Fig. 3) is used to decide the permanent channel on which fixed interface has to operate. The node tries to select the "best" channel based on information included in tables *NeighborTable* and *ChannelUsageList*. By the best channel, we mean the channel that is being less used by the 1-hop neighbors.
b. Periodically, every node broadcasts a BEACON message on each channel to notify neighbors about its currently fixed channel. All nodes that receive a BEACON message will add/update an entry in their *NeighborTable* (1-hop neighbor table) for the sending neighbor, with the objective of maintaining fresh information about all neighbors.
c. Eventually, a node is allowed to change its fixed channel, when the channel becomes too crowed. For that, the node will have to monitor its *ChannelUsageList*. If the node decides to change its fixed channel (to the channel with the least load), it will have to broadcast the information through a BEACON

message on all orthogonal channels, so that the other nodes can update their data structures (*NeighborTable* and *ChannelUsageList*).

3.2. COUNT_REQUEST and COUNT_REPLY Messages

Each node in the network transmits COUNT_REQUEST and COUNT_REPLY messages to its neighbors. The COUNT_REQUEST and COUNT_REPLY messages are a modified version of the one used in the implementation of the algorithms proposed in [26]. COUNT_REQUEST and COUNT_REPLY messages have the same Protocol Data Unit (PDU) and are composed of 10 fields (see Fig. 4).

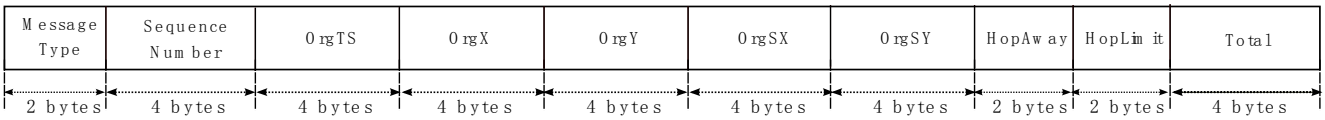

Figure 4. *COUNT_REQUEST and COUNT_REPLY messages.*

The field *Message Type* can be either 0 or 1. It is used to identify the type of message. A value of 0 is for a COUNT_REQUEST, while 1 is for a COUNT_REPLY. *Sequence Number* is used to match requests with replies and to distinguish between different requests (COUNT_REQUEST). *OrgTS* (Originator Timestamp) is set by the originator when it sends a COUNT_REQUEST message. It is a timestamp taken by the originator at the moment of sending the COUNT_REQUEST message and is aimed to control out-of-date messages and replay attacks. (*OrgX*, *OrgY*) is the position of the originator at the moment of sending the COUNT_REQUEST message. (*OrgSX*, *OrgSY*) is the speed of the originator at the moment of sending the COUNT_REQUEST message. The originator and the nodes broadcast COUNT_REQUEST messages along with the argument *HopAway* which represents the number of hops-away the receiver of the COUNT_REQUEST message is from the originator. The originator, which starts the process, must specify a value of *HopAway* equal to 1. Each node that rebroadcasts the message will select the smallest *HopAway* received up to now and will increment this field by 1. *HopLimit* is a way to control how far away COUNT_REQUEST messages are propagated from the originator toward the other nodes. It delimits the counting range. The field *Total* is filled with the number of nodes counted up to now. In COUNT_REQUEST messages, it is always equal to 0. Before sending a COUNT_REPLY message, a node must update this field according to the COUNT_REPLY messages received so far, and add 1 to the sum that represents itself.

3.3. Additional Information About the Algorithm

The originator first sends three messages of type COUNT_REQUEST as broadcast with a *HopAway* equal to 1 (separated by INT_BTW_REQ). When a node receives the first COUNT_REQUEST message, it will do the following after (see Fig. 1):

- RBCT_TIME: the node rebroadcasts the COUNT_REQUEST with a *HopAway* equal to the minimum *HopAway* received by this time +1. The node also registers the ID of the node that sent the COUNT_REQUEST with the minimum *HopAway*, in variable *nodeToGoBack*.
- RBCT_TIME + 1 * INT_BTW_REQ: the node rebroadcasts the previous COUNT_REQUEST message (because of possible collision with the first COUNT_REQUEST sent). If required, the fields of the COUNT_REQUEST message are updated. Also note that as specified in Section 5.1, the second rebroadcast of the COUNT_REQUEST is optional.
- RBCT_TIME + 2 * INT_BTW_REQ: the node rebroadcasts the previous COUNT_REQUEST message (because of possible collision with the first and second COUNT_REQUEST sent). If required, the fields of the COUNT_REQUEST message are updated. Also note that as specified in Section 5.1, the third rebroadcast of the COUNT_REQUEST is optional.
- REQ_TO: If the node did receive COUNT_REPLY messages, then the node computes the total of nodes based on variable *Total* received in COUNT_REPLY messages (+1 to represent itself in the total count) and sends the result to the closest node in the path to the originator, as a unicast message. If after REQ_TO the node did not receive any COUNT_REPLY message, then it generates a COUNT_REPLY with *Total* equal to 1 (this 1 represents itself) and sends it to the closest node in the path to the originator, as a unicast message.
- DELAY_ASYNC: If the node receives a delayed COUNT_REPLY message, then it recalculates the total number of nodes based on variable *Total* received in the COUNT_REPLY messages (+1 to add itself to the total count) and then sends the result to the closest node in the path to the originator, as an asynchronous COUNT_REPLY message.

4. Environments and Scenarios of Simulation

The algorithm that we propose in this paper was simulated with OMNeT++/INET. OMNeT++[1] is an open source, C++ based, multiplatform (Windows, MacOS, and Unix), discrete event simulator for modeling any system composed of devices interacting with each other. Its Graphical User Interface (GUI) is one of its main strengths, because through its GUI, users can create NED files (a description language to

[1] http://www.omnetpp.org

define the structure of the model) and inspect the state of each component during simulations. It has been used to model and simulate communication networks, operating systems, hardware architectures, distributed systems, and so on.

The fundamental ingredients of OMNeT++, that distinguish it from other simulators, are its object-oriented component architecture and message passing mechanism. Components (modules) are programmed in C++, and then are assembled into larger components using a high-level language (NED), which allow users to describe and define the structure of the model. For simulations of data networks, OMNeT++ relies on external extensions, such as INET. The INET Framework is an open-source communication network simulation package for the OMNeT++ simulation environment. It contains models for several wired and wireless networking protocols, including UDP, TCP, SCTP, IP, IPv6, Ethernet, PPP, 802.11, MPLS, OSPF, and many others.

We chose OMNeT++/INET because of its powerful GUI that facilitates the traceability and debugging of simulation models, by displaying the network graphics, animating the message flow and letting users peek into objects and variables within the model. Also, it is a very active project with many models that are constantly updated, making OMNeT++ a good candidate for both research and educational purposes [31].

For all the experiments, we selected WiFi (IEEE 802.11g) for the wireless communication standard, with a bitrate of 54 Mbps. The free space propagation model was chosen for path loss. Different scenarios were simulated where nodes are randomly distributed over a rectangular area. The random waypoint mobility model was selected to reflect the most general scenario of node movements.

5. Performance Study of the Simulations

In this section, we study the performance of the multi-interface multi-channel algorithm for counting nodes using wireless communications. We present details of the simulations that were executed in four types of scenarios: (1) in scenarios where both nodes and originator are stationary, (2) in scenarios with stationary nodes and mobile originator, (3) in scenarios with mobile nodes and stationary originator, and (4) in scenarios where both nodes and originator are mobile. Additionally, we executed simulations in scenarios where we varied the size of the rectangular area where the nodes and the originator are placed and can move. We compare the results of our new experiments with the ones obtained with the algorithm described in [26], reporting metrics such as: (1) counting accuracy, (2) response time to complete the counting, and (3) number of control messages required (COUNT_REQUEST and COUNT_REPLY messages sent by the nodes during the counting process).

5.1. Scenarios with Stationary Nodes and Stationary Originator

As with the proposed algorithm in [26], we performed various simulations for stationary scenarios, that is where the speed of the nodes and the originator are equal to 0 mps (meters per second). The nodes were randomly positioned in a squared scenario of 800m x 800m and the originator located in the middle of the squared scenario, starting the counting with a value of *HopLimit*=3.

The aim of our first experiment is to show the importance of improving the number of COUNT_REQUEST messages that a node should send to its neighbor nodes (which is not always equal to 3). Depending on the number of COUNT_REQUEST messages actually received from its neighborhood, a node can decide not to rebroadcast more COUNT_REQUEST messages depending on condition (2), since the node has determined that a considerable number of COUNT_REQUEST messages have already been sent in its neighborhood.

$$\text{ThrCountReq}*(3*\text{TotalNeighborNode}) >= \text{NumRecvMsgNode} \quad (2)$$

In condition (2), the parameters are:

ThrCountReq: Threshold that control the number of COUNT_REQUEST messages sent by the actual node.

TotalNeighborNode: Total number of 1-hop neighbors of the actual node.

NumRecvMsgNode: Number of COUNT_REQUEST messages received up-to-now by the actual node.

Table 2 shows the results that we obtained when varying the total number of stationary nodes (10, 30, 50, 100, 150, 200, and 250) and the value of *ThrCountReq*. The results in Table 2 are represented as values *a*/*b*/*c*/*d*, where *a* denotes the number of nodes that are within the scope of the originator using multihop routing (i.e., nodes that should be counted), *b* the number of nodes actually counted by the algorithm proposed in this paper, *c* the associated response time for counting, and *d* the total number of control messages (COUNT_REQUEST and COUNT_REPLY) sent by the nodes during the counting. We can observe that for low values of *ThrCountReq*, the algorithm performs a counting of nodes quite acceptable with a low response time and a small number of control messages. On the other hand, for high values of *ThrCountReq*, the algorithm performs a more accurate counting with an acceptable response time, but with a greater number of control messages sent by the nodes. Choosing the correct value for this parameter is a trade-off between an acceptable response time, a low number of control messages sent to by nodes, and a high accuracy of counting, so that application developers will have to select this parameter according to their needs.

Table 2. *Importance of the number of COUNT_REQUEST messages sent by nodes.*

ThrCountReq	Total Number of Stationary Nodes						
	10	**30**	**50**	**100**	**150**	**200**	**250**
0.1	7/7/2.67s/19	30/25/2.81s/39	50/20/2.55s/20	100/84/3.62s/109	150/132/2.82s/225	200/158/3.57s/329	250/188/3.64s/391
0.2	7/7/2.38s/19	30/25/2.81s/37	50/39/2.42s/55	100/98/3.43s/165	150/140/3.51s/265	200/156/3.55s/376	250/189/3.66s/500
0.3	7/7/2.38s/19	30/25/2.60s/37	50/49/3.10s/88	100/98/3.45s/182	150/145/3.51s/324	200/172/3.55s/431	250/198/3.53s/552
0.4	7/7/2.38s/19	30/25/2.26s/43	50/43/2.55s/73	100/99/3.69s/233	150/134/3.58s/380	200/160/3.55s/501	250/180/3.53s/637
0.5	7/7/2.67s/20	30/26/2.60s/48	50/46/2.53s/82	100/97/3.50s/266	150/133/3.20s/420	200/177/3.47s/615	250/194/3.98s/701
0.6	7/7/2.67s/20	30/25/3.45s/44	50/47/2.55s/101	100/98/3.32s/297	150/138/3.58s/441	200/177/3.47s/615	250/185/3.93s/772
0.7	7/7/2.67s/21	30/26/3.41s/51	50/50/2.55s/123	100/97/3.39s/305	150/138/3.29s/483	200/179/3.46s/663	250/177/3.48s/820
0.8	7/7/2.67s/20	30/28/2.67s/56	50/47/2.58s/127	100/94/3.33s/334	150/133/3.58s/497	200/172/3.51s/703	250/210/3.61s/877
0.9	7/7/2.67s/20	30/30/2.70s/73	50/47/2.39s/131	100/96/3.34s/347	150/135/3.52s/509	200/169/3.46s/712	250/175/3.61s/901
1.0	7/7/2.67s/20	30/27/2.61s/76	50/48/2.42s/147	100/97/3.48s/349	150/130/3.51s/528	200/165/3.46s/730	250/177/3.53s/929

To make it easier, let call *Algorithm I* the algorithm presented in [26], and *Algorithm II* the algorithm proposed in this paper. In Fig. 5, 6, and 7 we compare the performance of *Algorithms I and II* in different stationary scenarios, in terms of the number of nodes counted, the associated response time to count the nodes, and the number of control messages sent, where we varied the total number of static nodes (20, 30, 40, 50, 100, 125, 150, 200, and 250) randomly placed in a squared scenario of 800m x 800m, and the propagation range of the nodes and the originator (100 m, 200 m, and 300 m). Based on the results of [26], we chose the following values for the parameters RBCT_TIME=0.2s, INT_BTW_REQ =0.2s, and *ThrCountReq* =0.3 (according to Table 2), with the purpose of getting a more accurate counting. The originator started the counting process with *hopLimit*=3. For each number of static nodes in Fig. 5, the results are divided into groups of three bars depending on the propagation range value, i.e., the three blue bars correspond to 100 m, the three purple bars to 200 m, and the three green bars to 300 m. Moreover, in these groups of three bars, the first bar indicates the number of nodes that are within the scope of the originator using multihop routing (i.e., nodes that should be counted), the second bar represents the number of nodes actually counted by *Algorithm I*, and the third bar shows the number of nodes actually counted by *Algorithm II*. For each number of static nodes in Fig. 6, the results are divided into groups of two bars depending on the propagation range value, i.e., the two blue bars correspond to 100 m, the two purple bars to 200 m, and the two green bars to 300 m. In each group of two bars, the first bar indicates the response time for *Algorithm I* and the second bar represents the response time for *Algorithm II*. Finally, for each number of static nodes in Fig. 7, the results are divided into groups of two bars depending on the propagation range value, i.e., the two blue bars correspond to 100 m, the two purple bars to 200 m, and the two green bars to 300 m. For each group of two bars, the first bar indicates the total number of control messages sent by *Algorithm I* and the second bar represents the total number of control messages sent by *Algorithm II*.

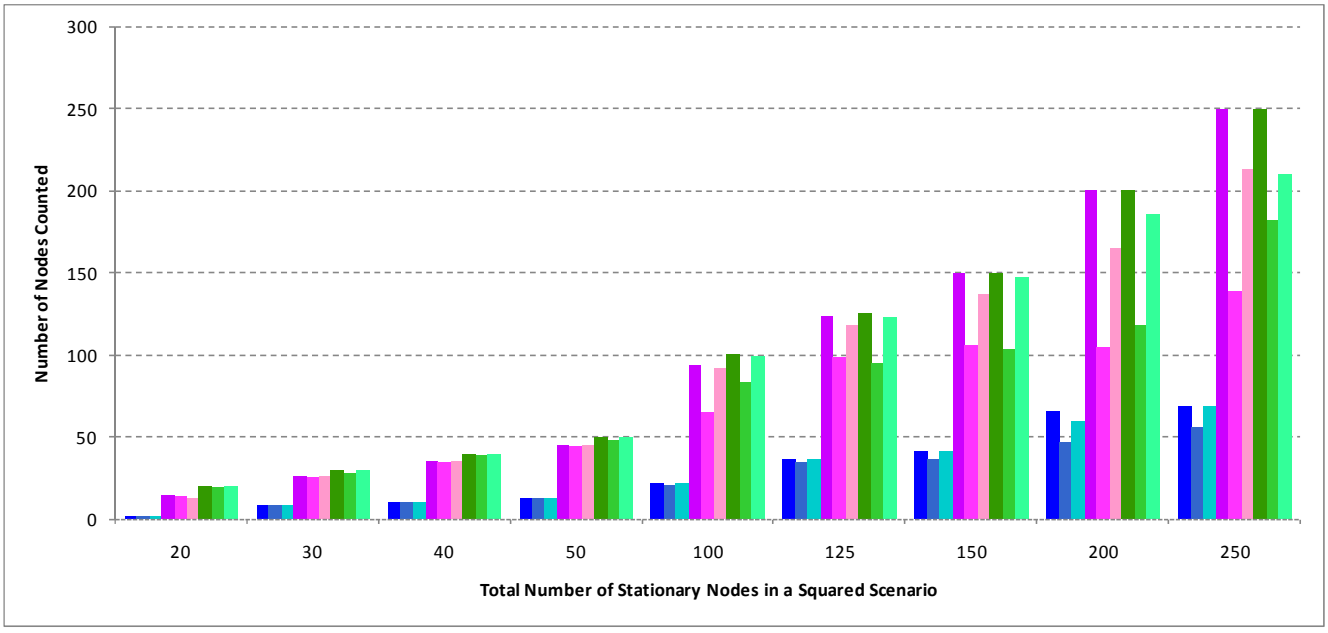

Figure 3. *Nodes counted in different scenarios with stationary nodes and stationary originator.*

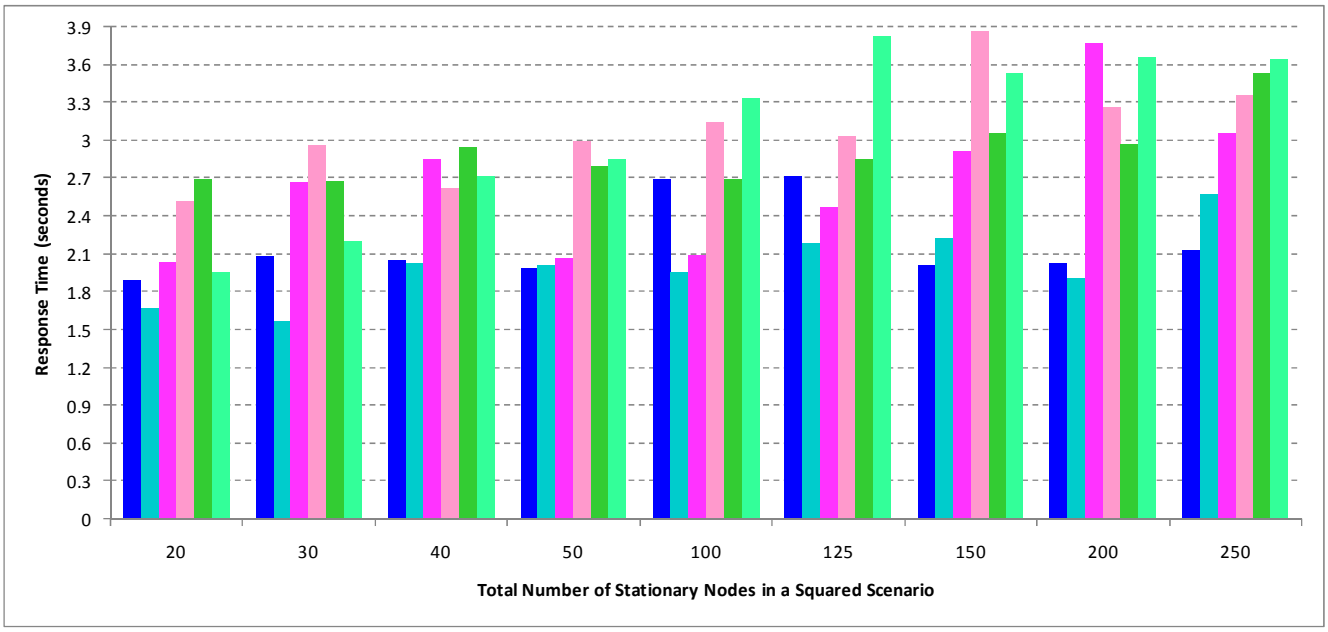

Figure 4. *Response time in different scenarios with stationary nodes and stationary originator.*

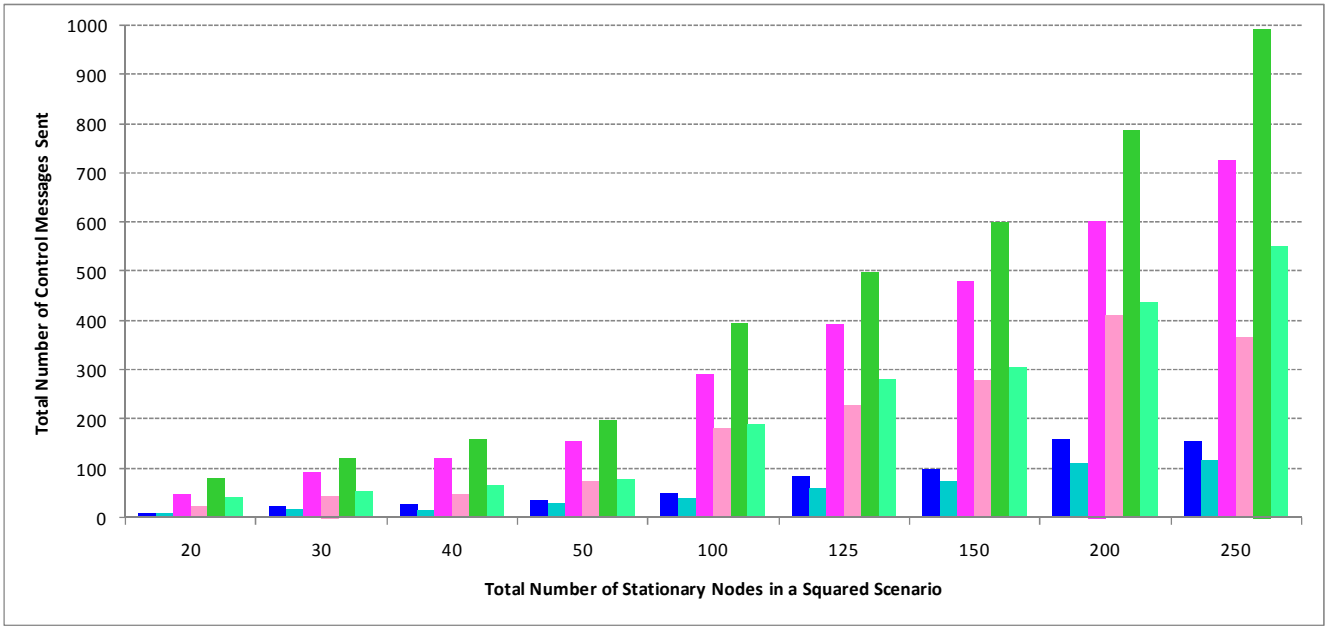

Figure 5. *Total number of control messages in different scenarios with stationary nodes and stationary originator.*

We can see that *Algorithm II* exceeds substantially *Algorithm I*, by performing a more effective counting, with a smaller number of control messages, and a very acceptable response time, especially in scenarios where there is a high density of nodes (100, 125, 150, 200, and 250). For example, Fig. 5 shows that for a scenario of 200 nodes and propagation range equal to 300 m (the three green bars), the origin should count 200 nodes. Now when we run the simulations, *Algorithm I* counted 118 nodes (see Fig. 5), instead of 200, with a response time of 2.97 s (see Fig. 6) and a total number of control messages equal to 785 (see Fig. 7), which is a good approximation. However, *Algorithm II* counted 186 nodes (see Fig. 5), being more effective, with a response time of 3.66 s (see Fig. 6), and a total number of control messages equal to 439 (see Fig. 7), which represents a minor counting error of 7.0% compared to 41% in *Algorithm I*. Regarding the simulations when the number of nodes is less than 100, *Algorithm II* is also very much more successful than *Algorithm I*.

5.2. Scenarios with Stationary Nodes and Mobile Originator

In this section, we analyze the performance of the two algorithms, in scenarios of stationary nodes and a mobile originator, in terms of the number of nodes counted, the associated response time to count the nodes, and the number of control messages sent. At the beginning of the simulations,

the static nodes were randomly positioned in the 800m x 800m squared scenario. The originator was initially placed at the center of the scenario, and was moving according to the random waypoint mobility model (RandomWPMobility), without making stops at the visited positions, i.e., with a wait time of 0s. Also, for these experiments, we chose values for the parameters RBCT_TIME=0.2s, INT_BTW_REQ=0.2s, and *ThrCoutReq* =0.3. The originator started the counting process with a *hopLimit*=3.

Fig. 8, 9, and 10 show the results of the simulations with 100 nodes, where we varied the speed of the originator (3, 5, 10, 20, 30, 40, 50, 60, and 70 mps) as well as the propagation range of the originator and the nodes (100 m, 200 m, and 300 m). For each speed in Fig. 8, the results are divided into groups of three bars depending on the propagation range value, i.e., the three blue bars correspond to 100 m, the three purple bars to 200 m, and the three green bars to 300 m. Besides, in these groups of three bars, the first bar indicates the number of nodes that should be counted by the originator, the second bar represents the number of nodes actually counted by *Algorithm I*, and the third bar shows the number of nodes actually counted by *Algorithm II*. For each speed in Fig. 9, the results are divided into groups of two bars depending on the propagation range value, i.e., the two blue bars correspond to 100 m, the two purple bars to 200 m, and the two green bars to 300 m. In each group of two bars, the first bar indicates the response time for *Algorithm I* and the second bar represents the response time for *Algorithm II*. Finally, for each speed in Fig. 10, the results are divided into groups of two bars depending on the propagation range value, i.e., the two blue bars correspond to 100 m, the two purple bars to 200 m, and the two green bars to 300 m. In each group of two bars, the first bar indicates the total number of control messages sent by *Algorithm I* and the second bar represents the total number of control messages sent by *Algorithm II*.

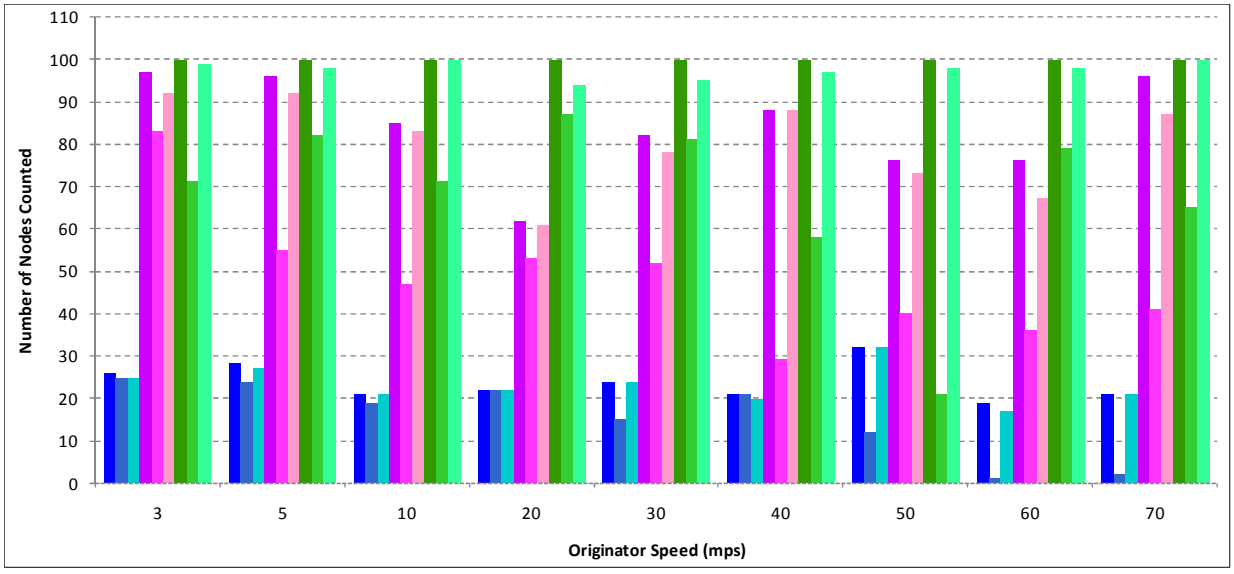

Figure 6. *Nodes counted in different scenarios with stationary nodes and mobile originator.*

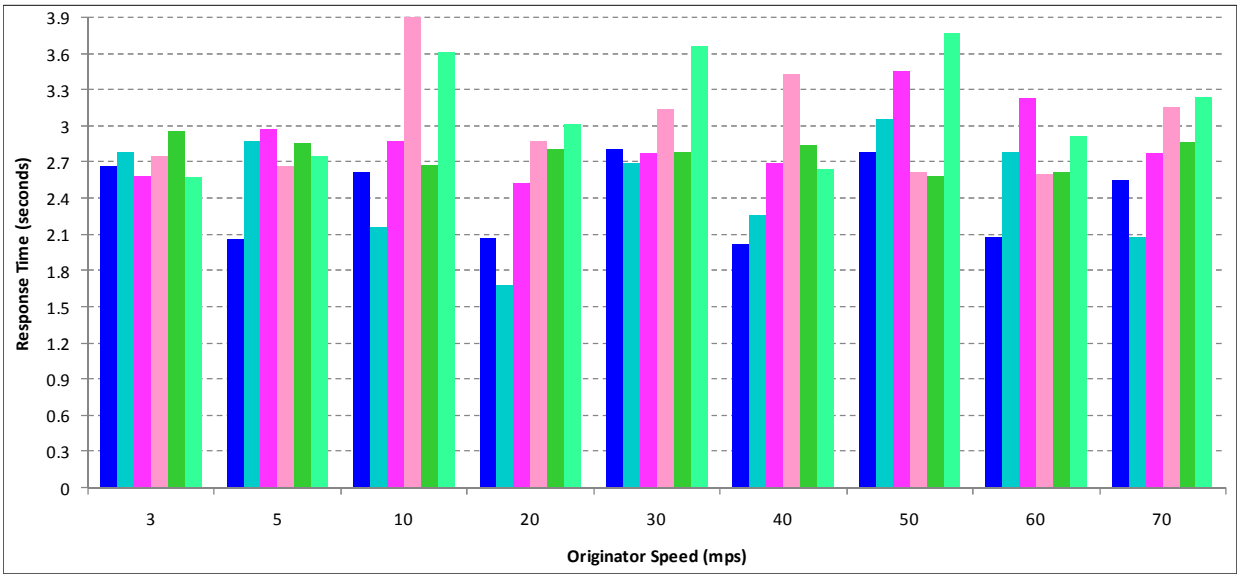

Figure 7. *Response time in different scenarios with stationary nodes and mobile originator.*

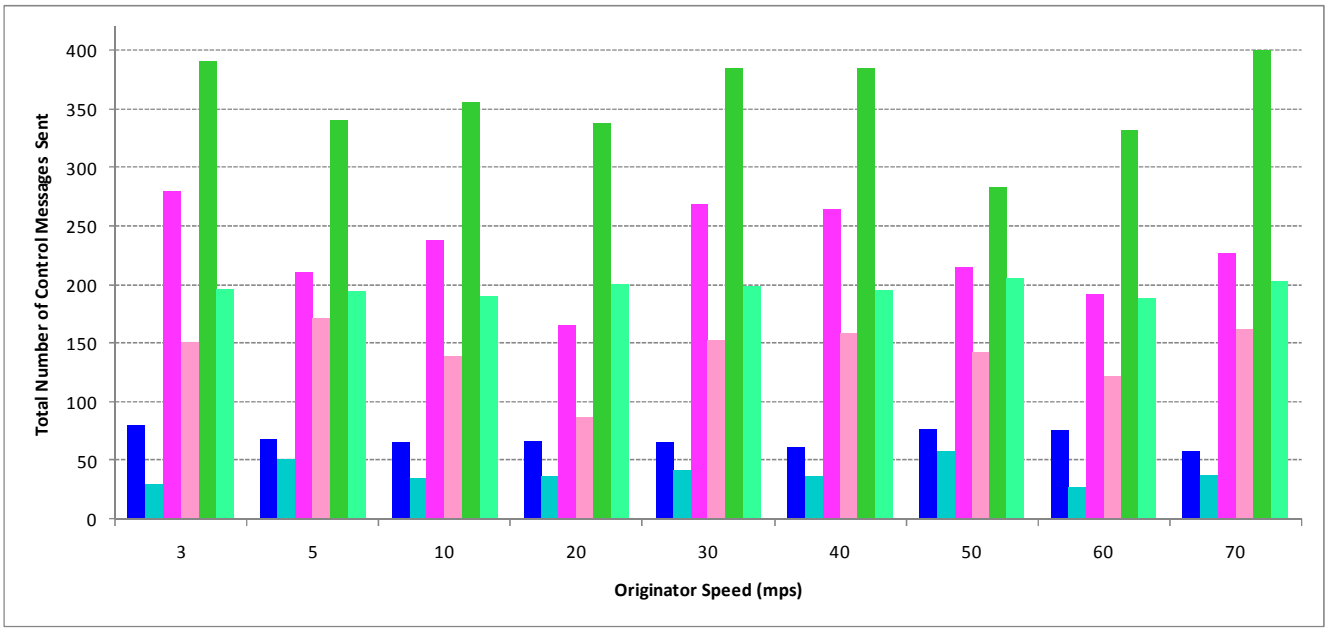

Figure 8. *Total number of control messages in different scenarios with stationary nodes and mobile originator.*

We can note that *Algorithm II* has a higher accuracy than *Algorithm I* in the counting of nodes, specifically when the originator moves at speeds greater than 30 mps, since at these speeds the accuracy of counting of *Algorithm I* degrades dramatically. For example, Fig. 8 shows that for a scenario where the originator moves at a speed of 40 mps (144 km/h) and with a propagation range equal to 300 m (the three green bars), the originator should count 100 nodes. Now when we run the simulations, *Algorithm I* counted 58 nodes (see Fig. 8), instead of 100, with a response time of 2.84 s (see Fig. 9) and a total number of control messages equal to 385 (see Fig. 10). However, *Algorithm II* counted 97 nodes (see Fig. 8), being more effective, with a response time of 2.65 s (see Fig. 9) and a total number of control messages equal to 195 (see Fig. 10), which represents a minor counting error of 3% in contrast to 42% of *Algorithm I*.

5.3. Scenarios with Mobile Nodes and Stationary Originator

In this section, we look at the performance of the algorithm when nodes are mobile, but the originator is static and placed in the center of an 800m x 800m squared area, based on the number of nodes counted, the associated response time to count the nodes, and the number of control messages sent by the nodes. For the movement of nodes, we used the random waypoint mobility model (RandomWPMobility) varying the speed of the nodes from 3 mps to 70 mps, with a wait time of 0s (time interval between reaching a target and moving toward a new one).

In Fig. 11, 12, and 13, we varied the propagation range of the nodes and the originator (100 m, 200 m, and 300 m). For all these scenarios, we chose 100 mobile nodes that were initially randomly placed in the 800m x 800m squared scenario with values for the parameters RBCT_TIME=0.2s, INT_BTW_REQ=0.2s, and *ThrCountReq*=0.3. The originator started the counting process with *hopLimit*=3. For each speed in Fig. 11, the results are divided into groups of three bars depending on the propagation range value, i.e., the three blue bars correspond to 100 m, the three purple bars to 200 m, and the three green bars to 300 m. Besides, in these groups of three bars, the first bar indicates the number of nodes that should be counted by the originator, the second bar represents the number of nodes actually counted by *Algorithm I*, and the third bar shows the number of nodes actually counted by *Algorithm II*. For each speed in Fig. 12, the results are divided into groups of two bars depending on the propagation range value, i.e., the two blue bars correspond to 100 m, the two purple bars to 200 m, and the two green bars to 300 m. In each group of two bars, the first bar indicates the response time for *Algorithm I* and the second bar represents the response time for *Algorithm II*. For each speed in Fig. 13, the results are divided into groups of two bars depending on the propagation range value, i.e., the two blue bars correspond to 100 m, the two purple bars to 200 m, and the two green bars to 300 m. For each group of two bars, the first bar indicates the total number of control messages sent by *Algorithm I* and the second bar represents the total number of control messages sent by *Algorithm II*.

According to the results shown in Fig. 11, 12, and 13, we can observe that we have high accuracy in the counting of nodes with *Algorithm II*. We can see that the best results are obtained with values of the propagation range equal to 200 m and 300 m. For example, Fig. 11 shows that for a scenario with a propagation range equal to 300 m (the three green bars) and a speed of nodes equal to 50 mps (180 km/h), the originator should count 100 nodes. Now, when we run the simulations, *Algorithm I* counted 62 nodes (see Fig. 11), instead of 100 nodes in 3.18 s (see Fig. 12), and a total number of control messages equal to 401 (see Fig. 13). However, *Algorithm II* counted 96 nodes (see Fig. 11), being much more effective,

with a response time of 3.75s (see Fig. 12) and a total number of control messages equal to 204 (see Fig. 13).

In Table 3, we chose a fixed propagation range of 250 m, but we varied the speed of the nodes (from 1 mps to 70 mps) and the number of nodes (10, 30, 50, 100, 150, 200, and 250) randomly placed in a squared scenario of 800m x 800m. The results for each speed of mobile nodes are divided in two rows (results for *Algorithm I* are in the upper rows and results for *Algorithm II* are in the lower rows). Cells of Table 3 are represented as values *a*/*b*/*c*/*d*, where *a* denotes the number of nodes that are within the scope of the originator using multihop routing (i.e., nodes that should be counted), *b* the number of nodes actually counted by the algorithm, *c* the associated response time to count the nodes, and *d* the total number of control messages sent during the counting.

Table 3. *Nodes counted when varying the speed of the nodes and the number of nodes (speed originator = 0 mps).*

Nodes Speed (mps)	Total Number of Nodes						
	10	**30**	**50**	**100**	**150**	**200**	**250**
1	7/7/2.89s/28	29/25/2.63s/86	50/46/2.84s/191	100/76/2.85s/370	150/116/2.98s/573	200/116/2.97s/779	250/115/3.76s/949
	7/7/1.63s/13	29/29/2.25s/57	50/50/2.44s/96	100/99/2.64s/178	150/141/3.38s/329	200/173/3.18s/432	250/224/3.79s/576
3	7/7/2.89s/32	30/27/2.61s/93	50/47/2.84s/194	100/83/2.87s/388	150/110/2.88s/591	200/143/2.96s/791	250/125/3.25s/961
	7/7/1.63s/13	30/30/2.54s/56	50/50/2.51s/91	100/98/2.95s/201	150/142/3.52s/338	200/174/3.60s/447	250/225/4.04s/664
5	10/10/2.02s/40	30/26/2.07s/96	50/49/2.84s/195	100/87/2.80s/394	150/97/3.69s/600	200/142/3.01s/788	250/135/3.71s/991
	10/10/1.63s/17	30/27/2.20s/36	50/50/3.09s/90	100/98/3.01s/195	150/146/3.55s/327	200/168/3.66s/437	250/200/3.97s/663
10	10/9/2.89s/37	30/23/2.78s/117	50/37/2.84s/200	100/89/2.81s/398	150/96/3.86s/597	200/150/3.83s/791	250/152/3.68s/997
	10/10/2.58s/16	30/30/2.52s/51	50/50/3.21s/89	100/98/3.51s/197	150/146/3.57s/300	200/170/3.54s/400	250/205/3.81s/695
20	10/10/2.89s/38	30/27/2.51s/120	50/44/2.84s/201	100/63/2.86s/397	150/80/3.34s/601	200/147/3.51s/794	250/138/3.88s/976
	10/10/2.72s/17	30/30/2.60s/54	50/50/2.72s/96	100/97/3.36s/201	150/143/3.69s/323	200/172/3.79s/443	250/203/3.98s/654
30	7/5/2.89s/29	30/20/2.08s/121	50/36/2.85s/195	100/64/2.93s/400	150/78/3.39s/588	200/109/3.89s/776	250/129/2.95s/982
	7/7/1.69s/14	30/29/1.69s/49	50/49/3.11s/112	100/92/3.32s/192	150/139/3.46s/317	200/169/3.99s/441	250/201/3.85s/635
40	8/7/2.89s/33	30/14/2.65s/108	50/37/2.89s/197	100/56/2.72s/391	150/90/3.59s/582	200/106/3.03s/794	250/133/3.96s/940
	8/8/1.66s/16	30/30/2.31s/54	50/50/2.96s/105	100/97/2.85s/210	150/136/3.88s/315	200/178/4.11s/485	250/211/4.05s/657
50	10/5/2.76s/37	30/18/2.73s/117	50/41/2.89s/199	100/53/2.95s/382	150/82/3.78s/585	200/107/3.89s/776	250/120/3.84s/979
	10/10/2.73s/18	30/29/2.51s/50	50/50/2.88s/86	100/94/4.47s/208	150/134/3.86s/319	200/174/3.79s/462	250/203/3.95s/623
60	10/6/2.76s/37	30/20/2.09s/111	50/34/2.84s/203	100/64/2.68s/391	150/64/2.89s/591	200/147/3.18s/767	250/115/3.95s/982
	10/10/1.65s/17	29/29/2.55s/62	50/49/2.95s/92	100/94/3.83s/195	150/137/3.91s/322	200/181/4.03s/492	250/197/4.15s/646
70	9/5/2.75s/36	30/23/2.69s/121	50/26/2.84s/194	100/39/3.83s/397	150/57/3.98s/576	200/94/3.71s/782	250/91/3.96s/982
	9/9/1.65s/14	30/30/3.79s/67	50/49/2.87s/98	100/89/3.01s/196	150/135/3.87s/327	200/169/3.64s/440	250/195/3.99s/647

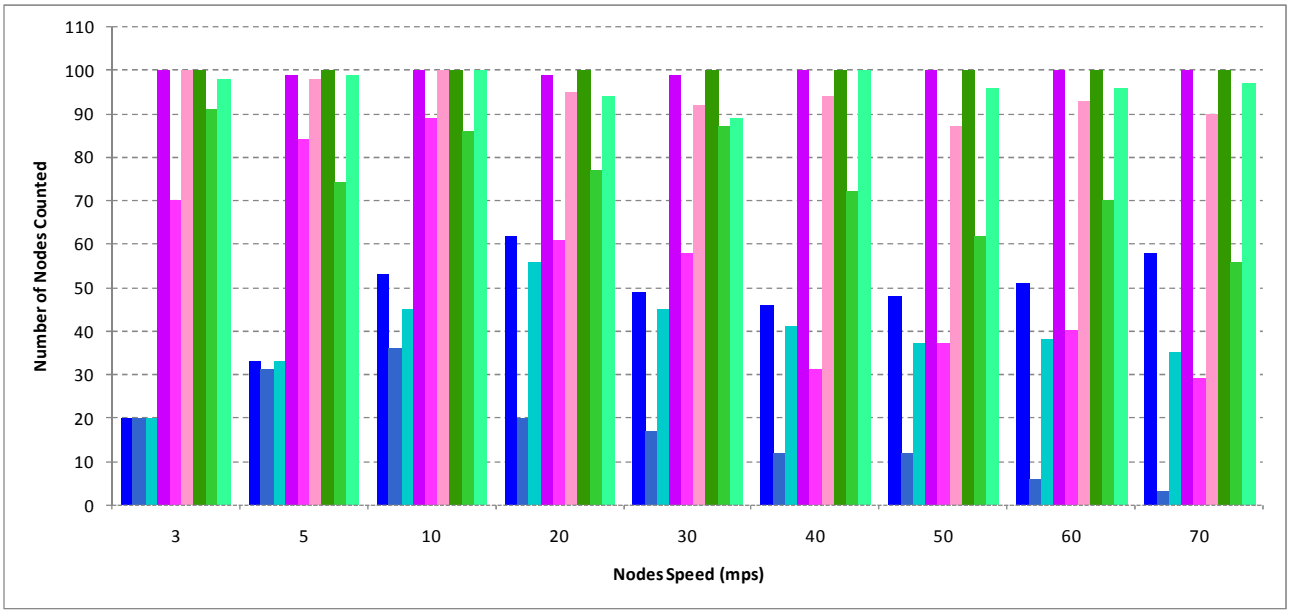

Figure 9. *Nodes counted in different scenarios with mobile nodes and stationary originator.*

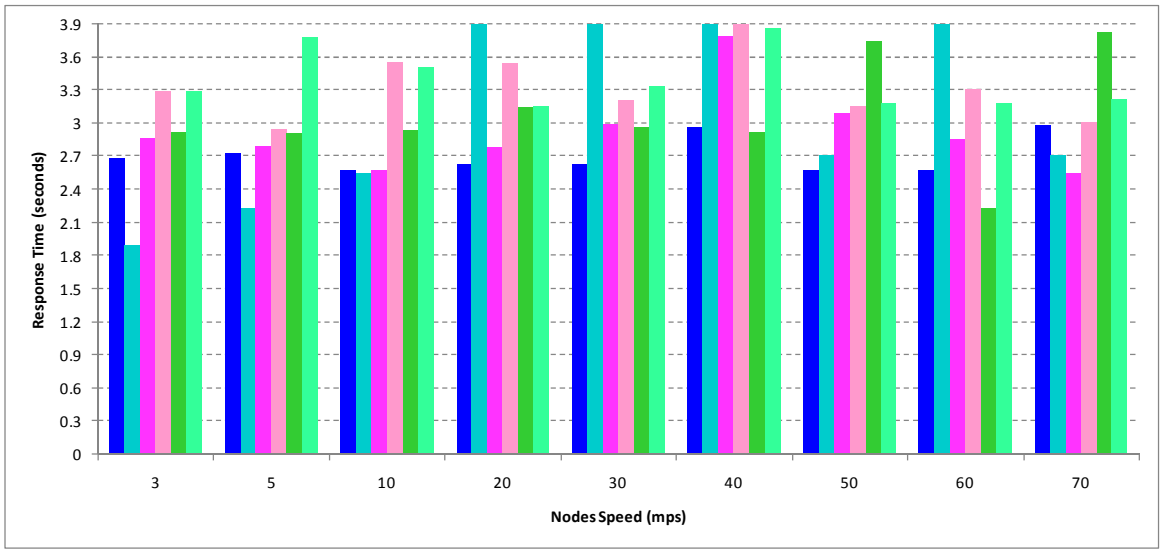

Figure 10. *Response time in different scenarios with mobile nodes and stationary originator.*

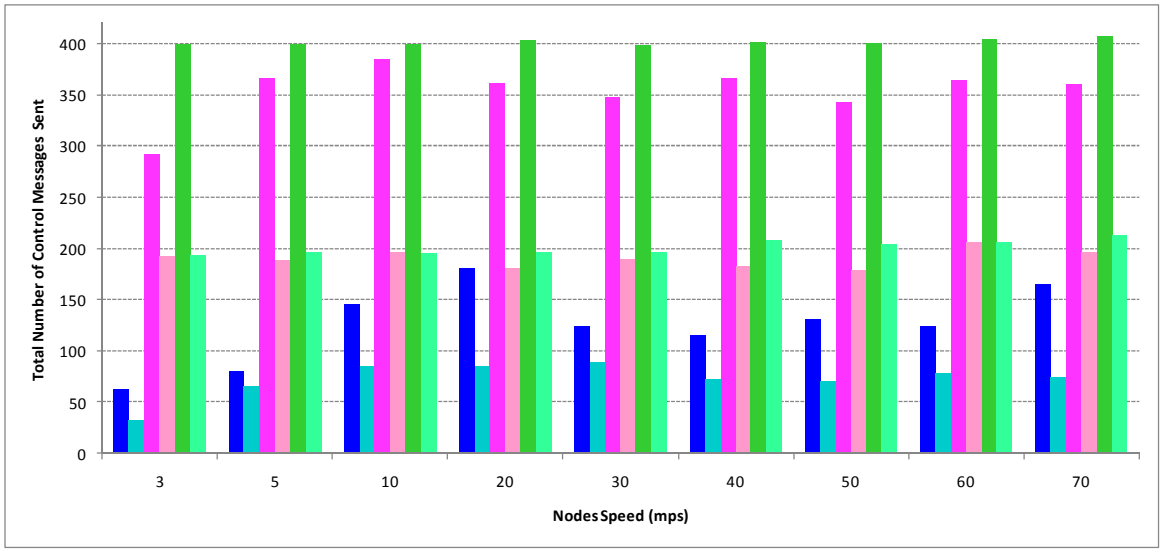

Figure 11. *Total number of control messages in different scenario with mobile nodes and stationary originator.*

Based on the results obtained in the Table 3, we can see that for a number of nodes up to 150, *Algorithm II* performs an effective counting even for very high speeds compared to *Algorithm I.* Now, for both algorithms when there is a high density of nodes (when the total number of nodes is equal to 200 and 250), the accuracy of the counting degrades, however it is still significantly more precise in *Algorithm II*. Additionally, the results obtained in question of response time and total numbers of control messages are substantially better for *Algorithm II*. For example, as reported in Table 3, for a scenario of 250 nodes and a speed of nodes equal to 50 mps (180 km/h), the originator should count 250 nodes. When we run the simulations, *Algorithm I* counted 120 nodes, with a response time of 3.84 s and a total number of control messages equal to 979, which are far below the results obtained in *Algorithm II*, with 203 counted nodes, a response time of 3.95 s, and a total number of control messages equal to 623.

5.4. Scenarios with Mobile Nodes and Mobile Originator

In this section, we report results for experiments of the proposed algorithm when both the nodes and the originator are mobile. At the beginning of the simulations, the nodes were randomly positioned in an 800m x 800m squared area, while the originator was initially placed at its center. Both, the nodes and the originator were moving according to the random waypoint mobility model (RandomWPMobility), without making stops at the visited positions, i.e., with a wait time of 0 s. For both, nodes and the originator, we chose RBCT_TIME=0.2s, INT_BTW_REQ=0.2s, and *ThrCountReq*=0.3. The originator started the counting process with a *hopLimit*=3.

In Table 4, we selected a fixed propagation range of 250 m, but we varied the speed of the nodes and the originator (from 1 mps to 70 mps), as well as the total number of nodes (10, 30, 50, 100, 150, 200, and 250). The results for each

speed of mobile nodes are divided in two rows (results for *Algorithm I* are in the upper rows and results for *Algorithm II* are in the lower rows). Cells of Table 4 are represented as values *a*/*b*/*c*/*d*, where *a* denotes the number of nodes that are within the scope of the originator using multihop routing (i.e., nodes that should be counted), *b* the number of nodes actually counted by the algorithm, *c* the associated response time to count the nodes, and *d* the total number of control messages sent during the counting.

Similarly to the simulations of Table 3, for a number of nodes up to 150, *Algorithm II* performs an effective counting even for very high speeds compared to *Algorithm I*. Now, for a number of nodes equal to 200 or 250, in both algorithms the accuracy of the counting degrades, however being substantially better in *Algorithm II*. For example, we can see in Table 4 that for a scenario of 250 nodes, and a speed of nodes and the originator equal to 40 mps (144 km/h), the originator should count 250 nodes. When we run the simulations, *Algorithm I* counted 129 nodes, with a response time of 3.26 s, and a total number of control messages equal to 943. However, *Algorithm II* counted 198 nodes with an associated response time of 3.98 s and a total number of control messages equal to 707. The results obtained in *Algorithm I* are by far outperformed by those obtained in *Algorithm II*.

Table 4. *Nodes counted when varying the speed of the nodes and the originator as well as the number of nodes.*

Speed (mps)	Total Number of Nodes						
	10	**30**	**50**	**100**	**150**	**200**	**250**
1	7/5/2.05s/22	29/29/2.55s/83	50/45/2.84s/191	100/78/2.72s/376	150/104/2.98s/555	200/140/2.85s/767	250/115/3.22s/934
	7/7/1.70s/14	29/29/2.88s/60	50/50/2.72s/83	100/98/3.45s/204	150/140/3.72s/306	200/193/4.23s/525	250/218/3.79s/588
3	7/7/2.03s/28	30/24/2.63s/93	50/48/2.91s/192	100/86/2.85s/382	150/88/3.0s/579	200/127/3.85s/770	250/132/3.52s/979
	7/7/1.70s/14	30/30/2.23s/57	50/50/2.52s/88	100/100/3.64s/206	150/144/3.84s/304	200/190/4.01s/549	250/217/3.64s/586
5	9/9/2.55s/30	30/28/2.56s/114	50/48/2.84s/194	100/78/2.90s/379	150/92/3.33s/585	200/149/2.49s/767	250/156/3.87s/976
	9/9/1.71s/14	30/30/1.75s/55	50/47/3.26s/79	100/99/3.57s/206	150/144/3.77s/308	200/190/3.96s/553	250/208/3.57s/530
10	10/9/2.81s/40	30/25/2.49s/117	50/46/2.94s/200	100/67/2.95s/376	150/80/2.90s/549	200/148/2.99s/698	250/159/3.69s/940
	10/10/1.64s/22	30/30/2.00s/58	50/50/2.46s/88	100/100/3.06s/202	150/138/3.57s/320	200/187/3.81s/538	250/203/3.47s/689
20	7/6/2.48s/33	29/27/2.11s/83	50/32/2.90s/182	100/54/2.80s/370	150/93/3.01s/567	200/144/3.02s/794	250/130/2.90s/829
	7/7/2.96s/17	29/29/3.90s/55	50/50/2.62s/86	100/97/3.09s/210	150/143/3.84s/311	200/184/3.95s/544	250/194/3.42s/654
30	9/9/2.72s/38	27/22/2.72s/101	50/40/2.67s/191	100/84/2.84s/330	150/77/3.00s/576	200/142/3.80s/773	250/153/3.51s/946
	9/9/2.47s/15	27/27/2.20s/51	50/49/2.15s/86	100/100/3.15s/196	150/139/4.01s/355	200/182/4.39s/552	250/193/3.60s/575
40	9/8/2.87s/30	30/17/2.78s/102	50/17/2.94s/161	100/68/3.0s/379	150/59/2.90s/540	200/108/3.01s/779	250/129/3.26s/943
	9/9/4.06s/18	30/30/4.80s/61	50/47/2.60s/75	100/94/3.64s/216	150/144/3.78s/352	200/175/3.95s/678	250/198/3.98s/707
50	10/5/2.77s/37	30/13/2.78s/110	50/39/2.84s/194	100/43/2.87s/364	150/96/2.92s/486	200/43/3.50s/679	250/92/2.96s/796
	10/10/2.42s/19	30/30/2.75s/62	50/49/2.88s/92	100/96/3.92s/192	150/141/3.95s/381	200/176/3.68s/653	250/195/4.12s/695
60	10/5/2.48s/34	30/16/2.56s/96	50/24/2.91s/185	100/64/2.83s/373	150/51/3.13s/558	200/78/3.04s/767	250/66/2.94s/892
	10/10/1.90s/21	30/30/2.47s/52	50/49/4.03s/78	100/94/4.37s/242	150/139/3.64s/365	200/168/3.89/625	250/192/4.60s/689
70	10/3/2.51s/40	30/17/2.65s/102	50/29/2.74s/182	100/41/3.22s/379	150/56/2.74s/561	200/36/3.51s/665	250/102/3.91s/964
	10/9/1.69s/21	30/30/2.69s/54	50/49/3.12s/95	100/92/3.83s/267	150/142/3.86s/377	200/173/3.91s/637	250/189/3.92s/672

Fig. 14, 15, and 16 show the results of the simulations for scenarios with 100 nodes, where we varied the speed of the nodes and the originator (from 3 mps to 70 mps), as well as their propagation range (100 m, 200 m, and 300 m). From there results, we can see that *Algorithm II* has a higher precision in the counting of nodes than *Algorithm I*, particularly in scenarios when the propagation range values are equal to 200 m and 300 m.

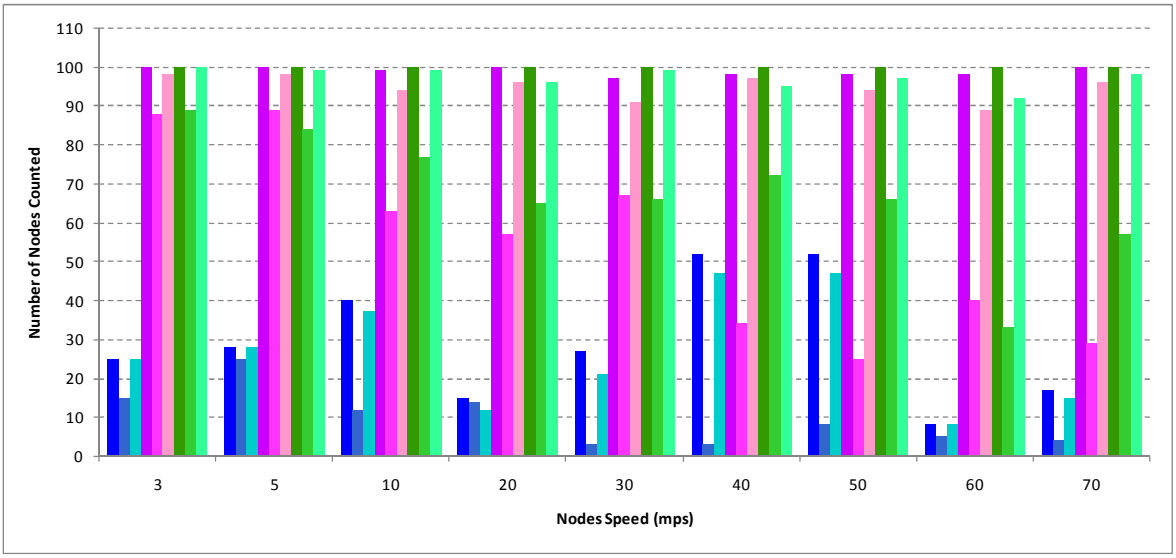

Figure 12. *Nodes counted in different scenarios with mobile nodes and mobile originator.*

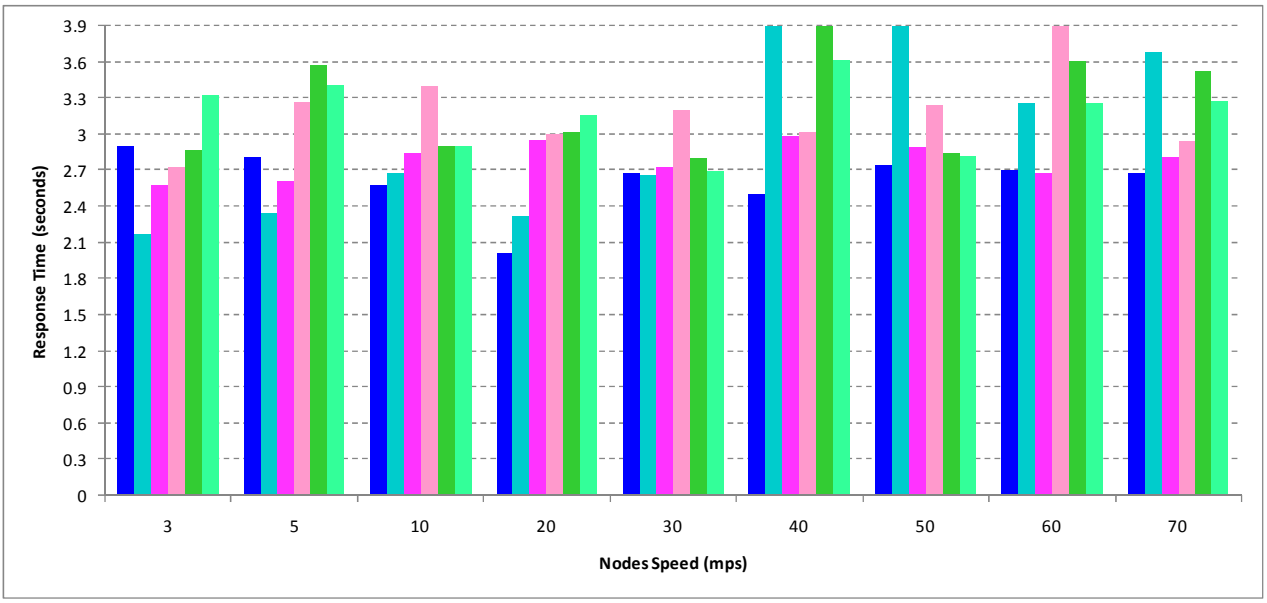

Figure 13. *Response time in different scenarios with mobile nodes and mobile originator.*

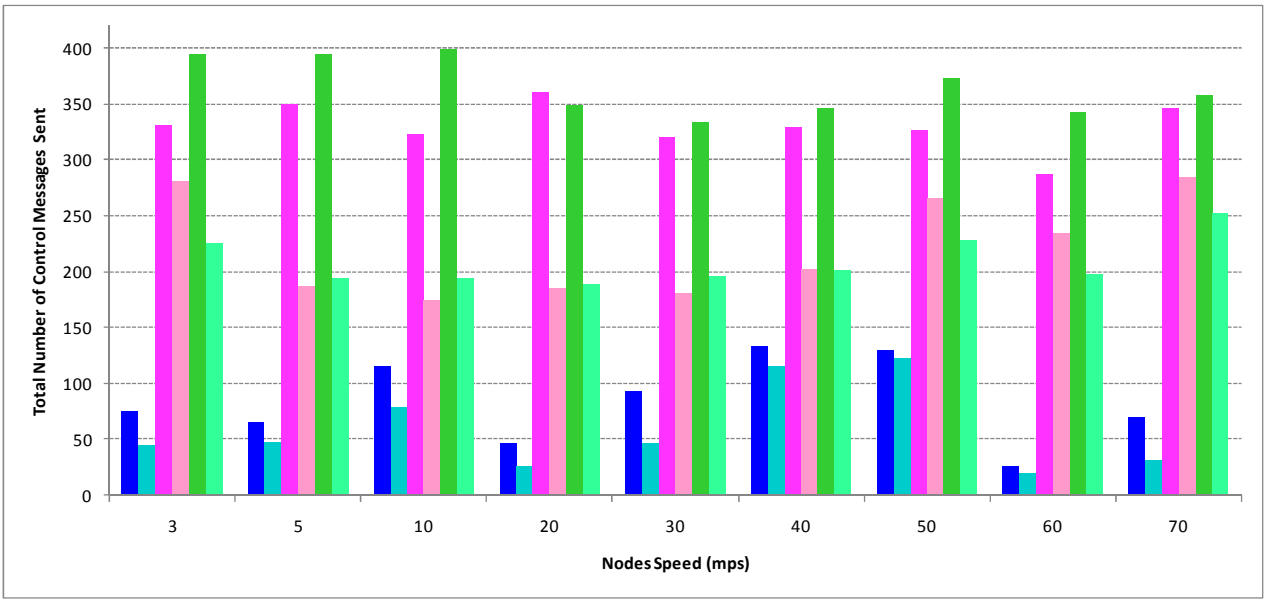

Figure 14. *Total number of control messages in different scenario with mobile nodes and mobile originator.*

5.5. Additional Experiments

In this section, we study the behavior of the proposed algorithm when we vary the size of the area where the nodes and the originator are placed, and therefore, check if it presents scalability issues. To do this, we performed some experiments by changing the size of the rectangular area with a density of 100 nodes/km^2 (see Table 5). At the beginning of the simulations, the nodes were randomly positioned in a rectangular area, while the originator was placed in its center. Both, the nodes and the originator were moving according to the random waypoint mobility model (RandomWPMobility), without making stops at the visited positions, i.e., with a wait time of 0s and a speed of 50 mps (180 km/h). For both, nodes and the originator, we chose RBCT_TIME=0.2s, INT_BTW_REQ=0.2s, and *ThrCountReq*=0.3, with a fixed propagation range of 300 m. The originator started the counting process with a *hopLimit*=3. In Table 5, from the first to the seventh column, we have: (1) the rectangular size of the simulation area, (2) the total number of nodes, (3) the number of nodes that are within the scope of the originator using multihop routing, (4) the number of nodes that the algorithms could count, (5) the associated response time, (6) the counting error in percent, and (7) the total number of control messages sent. Note that the last four columns have values represented as *a*/*b*, where *a* is for *Algorithm I* and *b* is for *Algorithm II*. In general terms, simulation results show that *Algorithm II* also performed well, in different rectangular areas, at high speeds, with a small number of control messages and an acceptable response time, being much more efficient than *Algorithm I*.

Table 5. *Nodes counted when varying the size of the rectangular area (mobil nodes and originator).*

Size of the Area	Total Number Nodes	Reachable Nodes	Nodes Count Algorithm	Response Time	Counting Error	Total Messages Sent
500m x 500m	25	25	21/25	2.11s/1.70s	16.00%/0.00%	100/41
500m x 750m	38	38	19/38	2.95s/2.72s	50.00%/0.00%	149/67
500m x 1000m	50	50	28/49	2.56s/2.46s	44.00%/2.00%	151/89
500m x 1250m	63	63	31/62	2.84s/2.59s	50.79%/1.58%	234/112
500m x 1500m	75	63	29/55	2.88s/3.08s	53.97%/12.70%	214/133
750m x 800m	60	59	22/59	2.81s/2.69s	62.71%/ 0.00%	198/130
750m x 1000m	75	72	28/70	3.62s/2.67s	61.11%/2.78%	203/139
750m x 1250m	94	91	56/85	2.79s/2.93s	38.46%/6.59%	298/162
800m x 800m	64	64	28/64	3.07s/2.97s	56.25%/0.00%	235/127
800m x 1000m	80	79	35/73	2.87s/2.78s	55.70%/7.60%	280/162
800m x 1250m	100	85	24/82	3.59s/3.51s	71.76%/3.53%	276/182
1000m x 1000m	100	98	55/86	2.84s/3.51s	43.88%/12.24%	296/187
1000m x 1250m	125	95	26/91	3.82s/4.73s	72.63%/4.21%	249/184
1250m x 1250m	156	122	44/101	3.06s/4.52s	63.93%/17.21%	338/228
1500m x 1500m	225	120	45/104	2.76s/4.45s	62.50%/ 13.33%	327/326
2000m x 2000m	400	241	53/198	2.82s/4.65s	78.00%/17.84%	665/672

6. Conclusions and Future Work

Multi-interface multi-channel wireless networks offer an aggregate bandwidth to users in the form of orthogonal channels. To take advantage of this possibility, new routing protocols, new channel assignment mechanisms, and in general new algorithms are required. In this research, we presented a novel multi-interface multi-channel algorithm to count nodes using wireless networks, based on ideas initially presented in [26]. The major enhancement in the new algorithm is the usage of two NICs in each nodes (only one NIC is required in [26]), with the intention to reduce interferences and collisions that occur between the counting PDUs (COUNT_REQUEST and COUNT_REPLY) and users' traffic. For that, one fixed channel is used to receive incoming control messages in the first interface. On the other hand, the objective of the second interface is the sending of outgoing control messages, and its current channel is switched accordingly to the fixed channel of the receiver node. Additionally, we implemented a discovery protocol where BEACON messages are broadcasted periodically, allowing nodes to discover their 1-hop neighbors and infer their actual position and speed at any time.

To validate and study the performance of our new algorithm, we performed simulations where we varied several parameters such as nodes' speed, total number of nodes, signal propagation range, size of simulation area, etc. The results that we obtained with the algorithm proposed in this paper outperformed the ones of our previous algorithm [26]. The analysis indicates that our new algorithm efficiently computes a total number of nodes very close to the real one, even in scenarios where the nodes are moving at very high speed with a small number of control messages and an acceptable response time.

As future work, we are interested in studying the application of our algorithm in the vehicular context (motorways, urban roads, rural roads, parkings, etc) using Wireless Access in Vehicular Environment (WAVE) [32], which could be used in applications to improve the safety and comfort of drivers and passengers. Also, many security issues will arise at the time of implementing the algorithm for a specific application, such as what nodes will have the authorization to start the counting process and how to enforce this policy to avoid flooding of the network or DoS (Denial-of-Service) attacks. This is another direction that we are interested to explore as an extension of our work.

Acknowledgment

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] S. Kakumanu. "Algorithms and Protocols for Multi-Channel Wireless Networks". In Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the School of Electrical and Computer Engineering. Georgia Institute of Technology. December 2011.

[2] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. "Architecture and Evaluation of an Unplanned 802.11b Mesh Network". In Proceedings of the 11th Annual International Conference on Mobile Computing and Networking (MobiCom 2005), Cologne, Germany, September 2005, pp. 31–42.

[3] C. Lochert, B. Scheuermann, C. Wewetzer, A. Luebke, and M. Mauve. "Data Aggregation and Roadside Unit Placement for a VANET Traffic Information System". In Proceedings of the fifth ACM International Workshop on VehiculAr Inter-Networking, New York, NY, USA 2008, pp. 58–65.

[4] A. Raniwala and C. Tzi-cker. "Architecture and Algorithms for IEEE 802.11 based Multi-Channel Wireless Mesh Network". In Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2005). Vol. 3, Miami, FL, USA, March 2005, pp. 2223-2234.

[5] P. Bahl, A. Adya, J. Padhye, and A. Walman. “Reconsidering Wireless Systems with Multiple Radios”. ACM SIGCOMM Computer Communication Review. Vol. 34, No. 5, October 2004, pp. 39–46.

[6] S. Pollak and V. Wieser. “Interference Reduction Channel Assignment Algorithm for Multi-Interface Wireless Mesh Networks”. In Proceedings of the 22nd International Conference Radioelektronika 2012, Brno, Czech Republic, April 2012.

[7] B. Barekatain, M. Aizaini, A. Ariza, A. Triviño, and H. Ghaeini. “An Enhanced Multi-interface Multi-channel Algorithm for High Quality Live Video Streaming over Hybrid WMNs”. Turkish Journal of Electrical Engineering & Computer Sciences, 2014, In press.

[8] J. So and N. Vaidya. “Multi-Channel MAC for Ad Hoc Networks: Handling Multichannel Hidden Terminals using a Single Transceiver”. In Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing. New York, NY, USA, May 2004, pp. 222–233.

[9] A. Nasipuri, J. Zhuang, and S. Das. “A Multichannel CSMA MAC Protocol for Multihop Wireless Networks”. In Proceedings of IEEE Wireless Communications and Networking Conference (WCNC 1999). New Orleans, LA, USA, September 1999, Vol. 3, pp. 1042-1406.

[10] A. Nasipuri and S. Das. “Multichannel CSMA with Signal Power-based Channel Selection for Multihop Wireless Networks”. In Proceedings of the 52nd IEEE Vehicular Technology Conference (VTC), Boston, MA, USA, September 2000, Vol. 1, pp. 211–218.

[11] N. Jain, S. Das, and A. Nasipuri. “A Multichannel CSMA MAC Protocol with Receiver-based Channel Selection for Multihop Wireless Networks”. In Proceediongs of the 9th International Conference on Computer Communications and Networks (ICSN), Scottsdale, AZ, USA, October 2001.

[12] S.-L. Wu, C.-Y. Lin, Y. C. Tseng, and J. P. Sheu. “A New Multi-Channel MAC Protocol with On-Demand Channel Assignment for Multi-Hop Mobile Ad Hoc Networks”. In Proceedings of the 2000 International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN’00), Dallas/Richardson, TX, USA, December 2000.

[13] U. Lee, S. Midkiff, and J. Park. “A Proactive Routing Protocol for Multi-Channel Wireless Ad-Hoc Networks”. In Proceedings of the 2005 International Conference on Information Technology: Coding and Computing (ITCC’05). Las Vegas, NV, USA, April 2005, Vol. 2, pp. 710–715.

[14] P. Kyasanur and N. Vaidya. “Routing and Interface Assignment in Multi-Channel Multi-Interface Wireless Networks”. In Proceedings of IEEE Wireless Communications and Networking Conference (WCNC 2005), New Orleans, LA, USA, March 2005, Vol. 4, pp. 2051–2056.

[15] D. Yong K. Pongaliur, and X. Li. “Channel Allocation and Routing in Hybrid Multichannel Multiradio Wireless Mesh Networks”. IEEE Transactions on Mobile Computing. Vol. 12, No. 2, February 2013, pp. 206-218.

[16] P. Kyasanur and N. Vaidya. “Routing and Link-Layer Protocols for Multi-Channel Multi-Interface Ad Hoc Wireless Networks”. ACM SIGMOBILE Mobile Computing and Communication Review. Vol. 10, No. 1, January 2006, pp. 31-43.

[17] H. Mogaibel, M. Othman, S. Subramaniam, and N. Hamid. “Impact of the Hybrid Multi-channel Multi-interface Wireless Mesh Network on ETX-Based Metrics Performance”. Electrical Power Systems and Computers. Vol. 99, 2011, pp. 147-160.

[18] L. Minglu and F. Yunxia. “Design and Implementation of a Hybrid Channel-Assignment Protocol for a Multi-Interface Wireless Mesh Network”. IEEE Transactions on Vehicular Technology. Vol. 59, No. 6. July 2010, pp. 2986-2997.

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

[20] C. Chao-Ho, C. Yin-Chan, C. Tsong-Yi, and W. Da-Jinn. “People Counting System for Getting In/Out of a Bus based on Video Processing”. In Proceedings of the 8th International Conference on Intelligent Systems Design and Applications (ISDA 2008). Kaohsiung, Taiwan, November 2008, Vol. 3, pp. 565-569.

[21] K. Chi. “Moving Object Counting with an Infrared Sensor Network”. A Thesis Submitted to The Hong Kong University of Science and Technology in Partial Fulfillment of the Requirements for The Degree of Master of Philosophy in Computer Science and Engineering. Hong Kong, August 2007.

[22] J. W. Kim, K. S. Choi, B. D. Choi, and S. J. Ko. “Real-time Vision-based People Counting System for the Security Door”. In Proceedings International Technical Conference on Circuits/Systems Computers and Communications, Phuket, Thailand, July 2002, pp. 1416-1419.

[23] T. Marques, L. Thomas, S. Martin, D. Mellinger, I. Ward, D. Moretti, D. Harris, and P. Tyack. “Estimating Animal Population Density using Passive Acoustics”. Biological Reviews. Vol. 88, 2013, pp. 287-309.

[24] H. Gross, P. Hunel, N. Vidot, and E. Stattner. “Acoustic Counting Algorithms for Wireless Sensor Networks”. In Proceedings of the 6th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks (PE-WASUN ’09), New York, NY, USA, 2009, pp. 79-84.

[25] 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, MA, USA, June 2000.

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

[27] M. Contreras and E. Gamess. “Algoritmo para Contar Nodos en Redes Inalámbricas con Mensajes Retrasados”. Revista Venezolana de Computación. Vol. 1, No. 2, December 2014, pp. 63-71.

[28] 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, NV, USA, July 2011, 327–333.

[29] C.-Y. Li, A.-K. Jeng, and R.-H. Jan. "A MAC Protocol for Multi-Channel Multi-Interface Wireless Mesh Network using Hybrid Channel Assignment Scheme". Journal of Information Sciencie and Engineering. Vol. 23, No. 4, July 2007, pp. 1041-1055.

[30] C. Chereddi, K. Pradeep, S. Jungmin, and N. Vaidya. "Multi-Channel Mesh Networks: Challenges and Protocols". IEEE Wireless Communications. Vol. 13, No. 2. April 2006, pp. 30-36.

[31] A. Varga. "The OMNeT++ Discrete Event Simulation System". In Proceedings of the 15th European Simulation Multiconference (ESM'2001). Prague, Czech Republic, June 2001.

[32] IEEE Trial-Use Standard for Wireless Access in Vehicular Environments (WAVE) - Multi-Channel Operation. IEEE 1609.4. November 2006.