Hindawi Publishing Corporation
Mobile Information Systems
Volume 2015, Article ID 820401, 18 pages
http://dx.doi.org/10.1155/2015/820401

Research Article

Autonomic Obstacle Detection and Avoidance in MANETs Driven by Cartography Enhanced OLSR

Abdelfettah Belghith,[1] Mohamed Belhassen,[2] Amine Dhraief,[2] Nour Elhouda Dougui,[2] and Hassan Mathkour[1]

[1]*College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia*
[2]*HANA Research Laboratory, University of Manouba, 2010 Manouba, Tunisia*

Correspondence should be addressed to Abdelfettah Belghith; abdelfettah.belghith@hotmail.com

Received 16 April 2015; Accepted 12 July 2015

Academic Editor: Salil Kanhere

The presence of obstructing obstacles severely degrades the efficiency of routing protocols in MANETs. To mitigate the effect of these obstructing obstacles, routing in MANETs is usually based on the *a priori* knowledge of the obstacle map. In this paper, we investigate rather the dynamic and autonomic detection of obstacles that might stand within the network. This is accomplished using the enhanced cartography optimized link state routing CE-OLSR with no extra signaling overhead. The evaluation of the performance of our proposed detection scheme is accomplished through extensive simulations using OMNET++. Results clearly show the ability of our proposed scheme to accurately delimit the obstacle area with high coverage and efficient precision ratios. Furthermore, we integrated the proposed scheme into CE-OLSR to make it capable of autonomously detecting and avoiding obstacles. Simulation results show the effectiveness of such an integrated protocol that provides the same route validity as that of CE-OLSR-OA which is based on the *a priori* knowledge of the obstructing obstacle map.

1. Introduction

In mobile ad hoc networks (MANETs), nodes cooperate together to insure an infrastructure-less multihop communication between distant mobile nodes. The main advantages of MANETs consist in the rapidity of their deployment and their low-cost compared to infrastructure-based networks. However, their proliferation is still limited by the inefficiency of current routing protocols to properly handle nodes mobility while preserving the limited and valuable resources of the network [1–4]. Resource scarcity of both the mobile nodes and the wireless medium precludes the use of additional control messages as it will be at the expense of less data traffic [5].

MANET routing protocols face an inevitable trade-off between maintaining valid routes and preserving valuable network resources (e.g., nodes power, nodes compute resources, radio resources, etc.). These trade-offs are accentuated by the presence of obstructing obstacles standing within the network area, which will increase the network dynamics. In fact, a link failure between two neighbouring nodes does occur not only when these nodes leave the range of each other but also when the line of sight between them is obstructed by an obstacle. To alleviate the side effect of obstacles, the underlying routing protocol should either be aware of the obstacle map or be capable of computing routes that avoid and get around these obstacles [6, 7].

MANETs, by their nature and according to their purpose, should rely on efficient routing protocols that are able to autonomously detect the exact location of obstacles within the network area. This is indeed a challenging task as the requirement for additional signaling to detect and share obstacle locations and contours should be limited to its minimum so that resources are left for the effective transmission of data. In this paper, we propose an autonomous obstacle detection scheme that relies on the cartography enhanced OLSR [6, 8]. Our proposed scheme does not require any additional signaling overhead as it relies completely on the use of CE-OLSR signaling but induces some additional

computation time at each node that will be identified and investigated in this paper.

Nodes mobility is another challenge facing the design of an autonomous obstacle detection strategy. Some obstacle detection approaches mark nodes lying on the obstacle boundaries instead of detecting its exact location [9, 10]. After this marking phase, the routing process considers this piece of information to avoid selecting paths passing through marked nodes. However, node mobility will rapidly stale the node marking process. To efficiently handle node mobility, such marking based approaches should significantly increase their signalling overhead and consequently consume valuable network resources otherwise left for effective data communication.

In this paper, we show that every node in the MANET can dynamically and autonomously infer the obstacle map without using either a dedicated technology (e.g. laser range finders, sonar, special optical/infrared sensors, etc.) or an additional signaling overhead. In fact, we propose a novel lightweight obstacle detection scheme entirely based on the original signalling of the cartography enhanced OLSR (CE-OLSR) protocol [8]. More precisely, we use the joint awareness of CE-OLSR about the link states and the network cartography to discover the pairs of nodes (position pairs) in the network that are unable to communicate due to obstructing obstacles. Subsequently, we process these position pairs to infer an approximation of the real obstacle boundary and its contour. The benefit of our proposal is fourfold. Firstly, our protocol is not based on any dedicated technology and hence can be easily integrated in any basic mobile node. Secondly, the proposed scheme requires no additional overhead since it uses the very same signalling of CE-OLSR. Thirdly, through dedicated metrics, we show that our scheme is able to accurately detect the real obstacle area and contour. Finally, using the proposed scheme, we reach almost the same routes validity of CE-OLSR with obstacle location awareness. Our intention here is not to outperform CE-OLSR with obstacle awareness (CE-OLSR-OA); rather we aim to relax the hard assumption considered in our previous protocol (i.e., CE-OLSR-OA) [6] that consists in imposing the availability of the obstacle map as an *a priori* knowledge in every node within the MANET area.

Obstacle detection and avoidance are well studied in static networks such as wireless sensor networks (WSNs). Nevertheless, to the best of our knowledge, there is no significant work done in the context of MANETs. Unlike stationary WSNs, MANETs are inherently and purposely mobile and dynamic. In this paper, we will show that on the contrary the mobility of nodes and the network dynamics constitute a real leverage for the obstacle detection efficiency. Some protocols rely on an *a priori* knowledge given to nodes such as obstacle map or street map to avoid the selection of links crossing obstacles during routing. But, in this paper, we propose an autonomous scheme that detects the contour of these obstacles using the underlying routing signalling and then dynamically integrate these obstacles' information to perform efficient routing.

The remainder of the paper is organized as follows. In Section 2, we review and discuss some relevant related works. In Section 3, we survey the functioning of the CE-OLSR protocol as well as the underlying core concepts of OLSR protocol. In Section 4, we detail our proposed obstacle detection scheme. In Section 5, we evaluate the performance of the proposed approach through extensive simulation tests. Some dedicated metrics are also proposed to assess the adequacy of our obstacle detection protocol. Furthermore, we study the computational complexity of the proposed obstacle detection scheme. In the last section, we conclude the paper.

2. Related Work

The existence of obstructing obstacles in the network area challenges the design of routing protocols in both static and mobile ad hoc networks. In the case of static ad hoc networks, link state routing protocols are less affected by the existence of obstacles compared to location-based protocols. Indeed, in link state routing protocols, weak or asymmetric links could be detected using some dedicated metrics such as the Expected Transmission Count [11] which measures the average number of data packet retransmissions for a specific link. But, in location-based routing protocols such as Greedy Perimeter Stateless Routing (GPSR) [12] or Greedy Face Greedy (GFG) [13], nodes do not have a global view about the network topology. Instead of relying on a global network topology, location-based routing protocols rely on the location of packet destination and that of neighbour nodes to perform routing decisions. In this class of protocols, the position of distant destination nodes is assumed to be readily available through some dedicated location services. So, these protocols do not include any cartography gathering strategy to collect the location of distant nodes.

Contrary to distant nodes, the locations of direct neighbours are known through the local signalling mechanism. Each node periodically broadcasts local control messages to inform its 1-hop neighbours about its position. Given the lack of the network topology in location-based routing protocols, nodes may route packets towards directions that lead to obstructing obstacles or voids in subsequent hops (>1-hop).

Location-based routing protocols like GPSR [12] or GFG [13] have commonly two operation modes: greedy mode and recovery mode. A given sender (or a forwarder) node operates in the greedy mode if it has a neighbour which is closer to the ultimate packet destination than itself. But when a sender (or a forwarder) detects that no one of its direct neighbours could bring the data packet closer to its ultimate destination, it infers that it is in the vicinity of a void or an obstacle. In such circumstances, it switches to the recovery mode. In recovery mode, data packets have to temporarily roll away from their destination using the right-hand rule to bypass the detected obstacle or void. Once data packets reach a node nearer to the destination than the node that initiated the recovery mode, this node resumes the greedy mode in order to avoid looping around the obstacle or the void.

The main advantage of GFG and GPSR consists in guaranteeing the delivery of data packets for static node and for sufficiently connected network even in the presence of obstructing obstacles. The main limitations of these two protocols reside in neglecting the optimality of the selected

routes. In fact, a given packet (routed in the direction of an obstacle or void) reaches the nodes at the obstacle boundary before being switched to a rescue path. Therefore, the resulting path is longer than the shortest possible path and this leads to a useless consumption of valuable resources.

In subsequent research work such as [10, 14–17], authors propose to decrease the selected path length by applying an early obstacle detection and avoidance scheme. For instance, in [10], authors make use of a local lightweight reputation mechanism to distributively find the possible shortest paths towards the destination while avoiding obstacles. The proposed reputation mechanism works as follows. Periodically, each node calculates its reputation based on its previous routing decisions. The reputation is proportional to the ratio between optimal and nonoptimal previous routing decisions. A given routing decision is termed optimal if greedy routing is used. Otherwise, it is termed nonoptimal. Each node broadcasts its reputation to its 1-hop neighbours. Therefore, nodes having bad reputation will be prevented from forwarding data packets during the greedy forwarding. Despite the ability of this obstacle avoidance scheme to progressively reduce the length of routing paths, it has some limitations. First of all, this scheme uses a unique trust value and thus it cannot handle multiple data streams. In fact, a given node N, which is not optimal for a given source/destination nodes pair, may be optimal for another pair of nodes. Subsequently, the path length of the second data stream may be uselessly increased. Furthermore, this scheme does not take into account node mobility. In dynamic networks, node reputation will rapidly change with the mobility of the different nodes. This will undoubtedly alter the efficiency of subsequent routing decisions. A part from the latter fact, these protocols ([10, 14–17]) succeed in avoiding obstacles while decreasing the length of resulting routes, but the real obstacle boundary or at least an approximation of this boundary remains unknown. This latter issue is addressed by Wang and Ssu in [9].

Authors of [9] proposed a scheme enabling detecting obstacles in WSNs. This solution approximates obstacle boundaries by marking the surrounding sensors nodes. In order to identify nodes that lie near the boundary of an obstacle (or void), authors introduced the notion of "crossing." The term "crossing" relates to the intersection point of the extremities of the sensing areas of 2 neighbouring nodes. A given crossing is termed covered if it stands in the sensing area of a third node. Otherwise, it is termed as an uncovered crossing. According to the proposed solution, if a node detects that one of its crossing is uncovered, it tags itself as a boundary node. This is argued by the fact that nodes standing near obstacle boundaries have usually an uncovered crossing. One of the main advantages of the proposed scheme consists in its low overhead, its robustness regarding ranging errors, and its general applicability. According to this solution, sensor nodes rely only on local information to identify obstacle boundaries. Furthermore, this scheme does not require any additional hardware or location-aware sensors to detect obstacles boundaries. However, the proposed scheme has several drawbacks and limitations. For instance, sensor nodes are assumed to remain static once they are deployed. As such, the proposed scheme is not suitable for sensor networks having some mobile nodes (sinks or regular nodes). In addition, authors did not explain how the detected obstacle information could be utilized to improve the operation of the network and did not mention how the obstacle information is shared among sensor nodes.

Contrary to static ad hoc networks, obstacle detection and avoidance strategies are less elaborated in the context of VANETs and MANETs. To the best of our knowledge, routing protocols rely on some a priori knowledge such as a street map or an obstacle map to be able to avoid obstacles. For instance, in the papers [7, 18–20] that treat routing in urban VANETs, authors relied on the awareness of street maps to route packets around obstacles.

In our prior work [6], we proposed an OLSR [21] based obstacle avoidance protocol named cartography enhanced OLSR with obstacle awareness (CE-OLSR-OA). We assumed that each node is aware of all obstacles standing within the network area. The joint awareness about network cartography and the map of obstacles made it possible to avoid links broken because of obstructing obstacles. In fact, two nodes are able to communicate together if and only if they are in the stable range of each other and the line of sight between them does not cross any obstacle. Nevertheless, according to the nature of MANETs, such information (obstacle map) is usually not known or not guaranteed to be available. Therefore, a more practical scheme allowing relaxing this hard assumption is required.

It is here worthy noting that none of the above routing protocols but [6] meet the requirements of an efficient obstacle detection scheme. An efficient obstacle detection algorithm should encompass some important features. Firstly, it has to cope with the scarcity of network resources by minimizing any required additional signalling. Secondly, it has to take into account the inherent unreliability of the wireless medium to be robust against control packet losses. Thirdly, to gain generality, an obstacle detection scheme uses neither dedicated devices during its operation nor any *a priori* knowledge about the network topography (obstacle/street map). Fourthly, to be suitable for MANETs, nodes mobility has to be supported. Finally, the obstacle detection scheme should deliver an accurate approximation of the obstacle boundary to insure a viable awareness.

In this paper, we propose an obstacle detection scheme that meets all the above requirements. Our proposed dynamic and autonomous scheme detects the obstructing obstacle and its contour and then integrates such information to perform an efficient routing. To infer the obstacle boundaries, it relies solely on the CE-OLSR [8], an enhanced version of OLSR, signalling composed of the links states information as well as the dynamically computed network cartography. In this work, we focus only on a single static convex or lightly concave obstructing obstacle. Further considerations and investigations have to be undertaken to handle the presence of several and highly concave moving obstructing obstacles.

3. Cartography Enhanced OLSR Protocol (CE-OLSR) Overview

CE-OLSR is an improvement of the well-known optimized link state routing (OLSR) protocol [21]. In this section, we start by reviewing the main features of OLSR. Then we highlight the impact of node mobility on OLSR. Finally, we survey the enhancements done by CE-OLSR to face problems resulting from nodes mobility.

OLSR protocol was first conceived to adapt link state routing to MANETs. As MANETs often suffer from the scarcity of radio resources, OLSR aims at reducing routing signalling to its minimum. To this end, OLSR implements two optimizations compared to basic link state routing protocols.

The first optimization consists in using two different control messages (Hello messages and Topology Control (TC) messages), with different frequencies, to track network topology. While Hello messages are used to track local topological change, TC messages are used to disseminate these local topological changes to distant nodes within the network. OLSR considers the splitting of the routing signaling into two types since local topological changes are more important to be tracked in timely manner than distant topological changes.

The second optimization introduced in OLSR consists in using Multipoint Relays (MPRs). MPRs are particular nodes that broadcast to the remaining nodes a subtopology of the network. Each node selects a subset of its 1-hope neighbours as MPRs to reach and cover its 2-hope neighbours. MPR selection strategies can be implemented in various ways according to the optimized parameters: nodes having the minimum ID [22], the coverage degree of 1-hop neighbours [23], and the bandwidth and/or delay of 1-hop and/or 2-hop links [24, 25]. Nodes that select a given MPR are termed as MPR-selectors. MPRs reduce the broadcast overhead of the OLSR signalling messages such as TC. Despite the success of OLSR in dealing appropriately with the scarcity of resources of ad hoc networks, its performance is highly affected by the increase of nodes mobility. We have demonstrated in our previous work [8] that once nodes start moving the routing performance of OLSR deteriorates. In the quest to mitigate this bad effect of mobility on OLSR performance, we proposed an enhanced routing protocol called CE-OLSR that is rather driven by the network cartography than the network topology. CE-OLSR relies on two main concepts: network cartography (nodes locations) and stability routing scheme. CE-OLSR uses the network cartography instead of link states to build a much precise network topology. We demonstrate in [2, 4, 8] that building a network topology based on nodes locations alleviates the mobility problem of proactive routing protocols. In CE-OLSR, we collect the network cartography using the very same original signalling of OLSR (no additional signaling is introduced). We assume that each node is aware of its location. Each node embeds in its Hello messages its position as well as the positions of its neighbours (collected from the received Hello messages). CE-OLSR also embeds nodes locations in TC messages by associating with each published link the position of the corresponding nodes. To enhance the validity of selected routes, CE-OLSR relies on a stability routing scheme [26] that avoids using weak links during the routing process. A given link is stated to be weak if it is established between two neighbours that are close to leave the range of each other. Essentially, route stability is accomplished by willingly underestimating the perceived network topology [2, 4, 26].

Cartography enhanced OLSR with obstacle awareness (CE-OLSR-OA) [6] is an enhanced version of CE-OLSR which avoids obstructing obstacles standing within the network area. CE-OLSR-OA requires an obstacle map to be known and initially fed into the system. More precisely, when building the network topology based on the collected network cartography (nodes locations), CE-OLSR-OA filters out all links that cross through the obstacles. It executes then a shortest path algorithm (Dijkstra) on the resulting connectivity graph to select a candidate gateway to each reachable destination. As we previously stated, the assumption regarding the availability of obstacle map challenges the usefulness of CE-OLSR-OA in practice for real ad hoc scenarios. In the following, we propose a dynamic and autonomous obstacle detection scheme that does not require such a hard assumption but rather is capable of detecting the obstacle residing within the network with a high accuracy and detects its boundary (contour) with high precision, yet it self integrates the resulting obstacle information in its routing decisions to avoid the obstacle and attain the same route validity as that of CE-OLSR-OA.

4. Materials and Methods: Obstacle Detection Scheme Description

Obstacles have several harmful effects on the functioning of MANETs as outlined in Section 3. In order to avoid obstacle side effects, MANETs should implement their own strategies to detect obstructing obstacles exact locations and boundaries. In this section, we present a novel lightweight obstacle detection strategy which natively takes into account the inherent constraints of MANETs. Firstly, it saves the valuable MANET resources as it does not introduce any new control traffic and solely relies on the exact CE-OLSR signalling. Secondly, as MANET wireless channel is notoriously unreliable, our strategy tolerates some amount of control packet losses. Finally, our proposed strategy is suitable for mobile environments. Indeed, it does not rely on any node tagging mechanism as it is done in stationary sensor networks. Instead of detecting nodes surrounding obstacles, our scheme identifies the location of the obstructing obstacle and its contour with a high accuracy.

Let us start by defining some terms used in the design and specification of our obstacle detection scheme.

4.1. Definitions

Definition 1 (obstacle information). Obstacle information is a position pair $([A_L, A_R])$ where two given neighbour nodes are unable to communicate because of an obstructing obstacle standing between them. A_L and A_R denote, respectively, the left and right extremity of the corresponding obstacle information (the obstructed link).

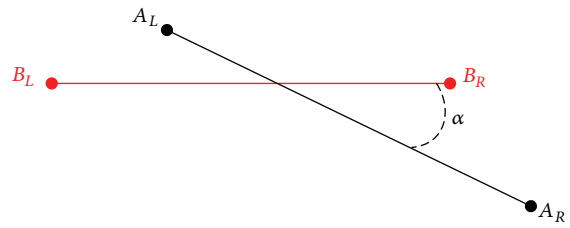


Figure 1: Inner angle between 2 given obstacle pieces of information.

Definition 2 (obstacle information accuracy). Obstacle information $[A_L, A_R]$ is considered more accurate than obstacle information $[B_L, B_R]$ if the positions forming its extremities are closer to the obstacle boundary than those of $[B_L, B_R]$.

Definition 3 (complementary obstacle information). Two obstacle pieces of information $[A_L, A_R]$ and $[B_L, B_R]$ are complementary if at least one of them could be used to enhance the accuracy of the other piece of information.

Definition 4 (obstacle information having similar directions). Two obstacle pieces of information $[A_L, A_R]$ and $[B_L, B_R]$ have similar directions if the inner angle α formed by their direct segments is less or equal to a fixed parameter called *MaxAngleAperture* as shown in Figure 1.

Definition 5 (obstacle information belonging to the same network region). Two obstacle pieces of information $[A_L, A_R]$ and $[B_L, B_R]$ belong to the same network region if and only if they satisfy the following two conditions. Firstly, the orthogonal projection of at least one extremity of one of these obstacle pieces of information (say $[A_L, A_R]$) on the line carrying the second one (say $[B_L, B_R]$) must be inside the segment representing the second obstacle piece of information. Secondly, the distance between the projected extremity and its orthogonal projection has to be less than the *MaxToleratedFarness* parameter.

Definition 6 (prunable obstacle information). Two given obstacle pieces of information $[A_L, A_R]$ and $[B_L, B_R]$ are prunable if they have similar directions and belong to the same network region.

4.2. Obstacle Detection Overview. Our obstacle detection strategy is composed of 4 steps (as depicted in Figure 2): (1) identification of noncommunicating positions, (2) pruning, (3) filtering, and (4) concave hull construction. In order to sustain the clarity of the following description, we illustrate the output of each step by a dedicated figure.

In step (1), we infer the obstacle information. We represent obstacle information, according to the context, either by a line segment or by a pair of points. Figure 3 shows the obstacle information collected by an arbitrary node during the first 100 s of a performed simulation. During this time, a huge number of obstacle pieces of information are collected by all the nodes.

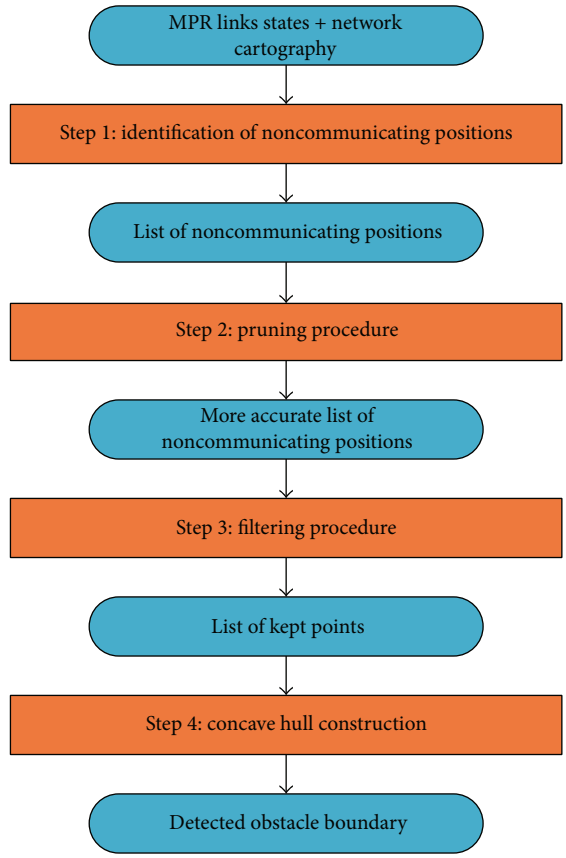


Figure 2: Building blocks of the proposed obstacle detection approach.

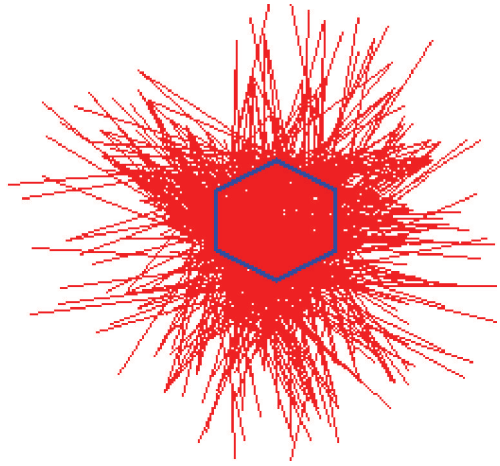


Figure 3: Step 1: obstacle information collected by an arbitrary node (simulation time = 100 s).

In step (2) of our obstacle detection scheme, we conceive a dedicated pruning approach. This approach aims at enhancing the accuracy of the collected obstacle information. Figure 4 depicts the obstacle information retained after performing this second step. By comparing Figure 4 with Figure 3, we notice that the obstacle information resulting from the second step is closer to the real obstacle boundaries

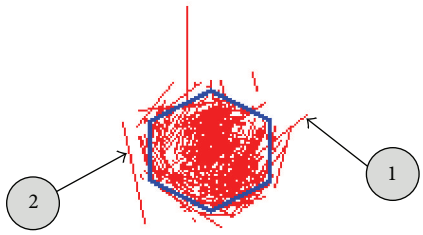


Figure 4: Step 2: obstacle information retained after the pruning step: (simulation time = 100 s).

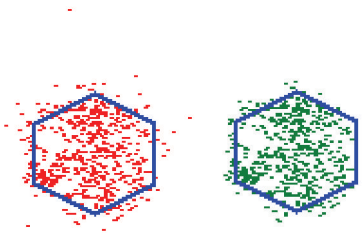


Figure 5: Step 3: (a) before filtering, (b) after filtering (simulation time = 100 s).

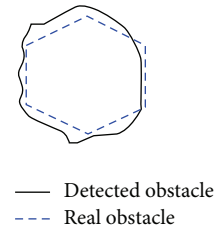
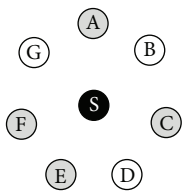


Figure 6: Step 4: concave hull construction (simulation time = 100 s).



TC_redundancy = 0	TC_redundancy = 2
A	A
C	C
E	E
F	F
	B
	D
	G

Content of TC_S

MPR node
MPR-selector of S
Simple node

Figure 7: The content of TC message generated by S in the absence of obstructing obstacle.

than the original one. Despite the success of the second step in enhancing the accuracy of collected obstacle information, the retained data might still include some inaccurate or erroneous obstacle information. Inaccurate obstacle information is generally caused by the absence of pruning opportunities (see Figure 4 arrow 1). In other words, one or both extremities of some obstacle information remain far from the obstacle boundary because of the lack of pruning opportunities with other collected obstacle pieces of information. Erroneous obstacle information is mainly caused by successive collisions of CE-OLSR control messages (see Figure 4 arrow 2).

Step (3) of our strategy eliminates inaccurate or erroneous obstacle information from the set of retained obstacle pieces of information. Figure 5 shows the result of this filtering step.

Finally, in step (4), we build the concave hull containing the unfiltered extremities of obstacle information. Figure 6 illustrates the obstacle boundary inferred by our obstacle detection scheme. Subsequent subsections provide the detailed operation of each of the aforementioned steps.

4.3. Step 1: Identification of Noncommunicating Positions (Obstacle Information). In this step, we use CE-OLSR signalling to detect positions pairs wherein two given neighbours are unable to communicate together because of obstructing obstacles standing between them.

When a node N detects that a given MPR node M has a neighbour V inferred from the network cartography rather than directly published in a TC message generated by M, it adds the positions of M and V to the list of noncommunicating positions pairs. This first step of our obstacle detection scheme is integrated into CE-OLSR protocol as follows. We slightly adjust OLSR default behaviour. In OLSR, when an MPR node sends a TC message, by default, it declares only its MPR-selector nodes, whereas, in our strategy MPR nodes send the entire neighbours list. Note that sending the entire neighbours list in TC messages does not violate the OLSR specification [21]. Adopting such an adjustment on TC messages allows the deduction of obstacle information. In fact, if a given MPR (S) includes only its MPR-selectors in its TC messages, namely, by putting TC_redundacy equal to zero, then the absence of a link state of a given neighbour (V) does not necessarily mean that (V) is not a neighbour of (S). For instance, in Figure 7, if we set TC_redundancy parameter to 0, (S) includes only its MPR-selector nodes (A, C, E, and F) in its TC (TC_S). In such a case, the absence of links towards the remaining neighbour nodes (B, D, and G) in TC_S does not mean that they are not neighbours of (S). We may only conclude that (B), (D), and (G) nodes are not MPR-selectors of (S).

However, when we impose sending the entire neighbour list in TC messages (i.e., TC_redundancy = 2), such a deduction becomes possible. For example, in Figure 7 all nodes within the transmission range of S are included in TC_S (see TC's content when TC_redundancy = 2). In Figure 8, the direct communication between (S) and (B) (resp., (S) and (C)) is obstructed by an obstacle. In such a case, (B) and (C) are not declared in TC_S even though they are in the transmission range of S. Subsequently, each node receiving TC_S could deduce that there is an obstacle between (S) and (B) (resp., (S) and (C)).

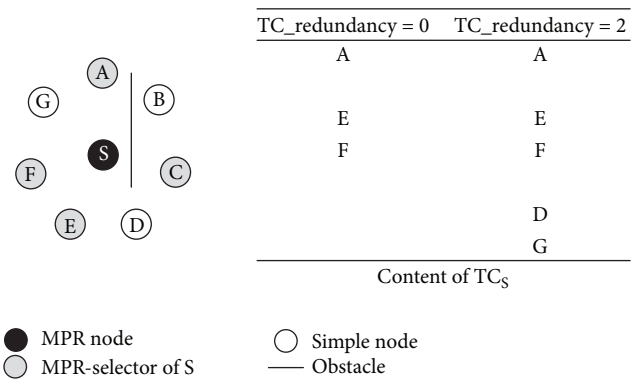


FIGURE 8: The content of TC message generated by S in the presence of obstructing obstacle.

Upon receiving a TC message (originally generated by a given MPR (S)), the receiver node (R) operates as follows. Let L_1 be the neighbours list published in the TC and L_2 the neighbours list of (S) known through the network cartography collected by the receiver node (R). For each node (V) existing in L_2 and not in L_1, node (R) has to add a new pair of positions (P_S, P_V) to the list of noncommunicating positions pairs. In order to sustain the robustness of our scheme toward control packet losses, we build L_2 using a *reduced transmission range* smaller than the real transmission range. Indeed, nodes newly entering the range of S and not yet published in (S) TC messages could be wrongly seen as obstructed by obstacles. We use a reduced transmission range in order to minimize such a false obstacle information detection. Using this scheme, each node progressively deduces the positions pairs for which neighbouring nodes are unable to communicate due to the obstructing obstacle. Over time, each node will therefore collect a sufficient list of noncommunicating positions pairs.

Before moving to the algorithmic details, let us focus on the first step complexity or execution time. Let N be the number of nodes and let M be the average number of MPR nodes (publishing TC messages). Let ObstacleDetectionPeriod be the periodicity of our obstacle detection scheme and TcPeriod be the periodicity of TC messages. In a time window equals to ObstacleDetectionPeriod, each MPR node generates N_{TC} TC messages, where N_{TC} is given by

$$N_{\mathrm{TC}} = \frac{\mathrm{ObstacleDetectionPeriod}}{\mathrm{TcPeriod}}. \quad (1)$$

For each received TC message, we need to compare L_1 to L_2. Let P be the maximum length of these lists. The worst overall computation time of this step is then

$$T = \Theta\left(N_{\mathrm{TC}} * M * P^2\right). \quad (2)$$

As the length of each list and the number of MPRs are both bounded by the number of nodes N, the overall computation time is then

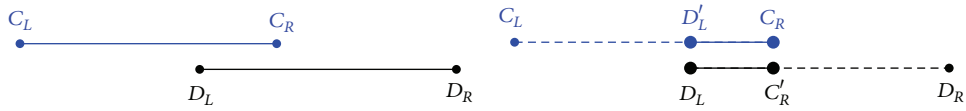
$$T = \mathcal{O}\left(N_{\mathrm{TC}} * N^3\right). \quad (3)$$

The collected obstacle information (i.e., the noncommunicating position pairs) should be mutually compared, processed, and pruned so that they are brought closer to the real obstacle boundaries (contour). This is achieved using a straightforward pruning procedure detailed next.

4.4. Step 2: Pruning Procedure. The pruning procedure is based on a simple complementarity concept that may exist between two given collected obstacle pieces of information. For instance, Figure 9(a) represents a case of complementary obstacle information. In this situation, a more accurate obstacle information pair could be inferred from the noncommunicating positions pairs ($[C_L, C_R]$ and $[D_L, D_R]$). In fact, from the first obstacle information ($[C_L, C_R]$), we deduce that the obstacle resides somewhere between C_L and C_R. Similarly, from the second piece of information ($[D_L, D_R]$) we conclude that there is an obstacle between D_L and D_R. Since, we assume that there is only one convex obstacle in the network area, it is necessarily located in the common area bounded by D_L and C_R. In this case, the left extremity of $[C_L, C_R]$ could be pruned up to D'_L (orthogonal projection of D_L on the line carrying $[C_L, C_R]$) and the right extremity of $[D_L, D_R]$ could be pruned up to C'_R (orthogonal projection of C_R on the line carrying $[D_L, D_R]$). That way, the original obstacle information pair could be replaced by a more accurate one ($[D'_L, C_R]$ and $[D_L, C'_R]$); see Figure 9(b). Similarly, using the defined complementary concept, in Figure 10(a), the obstacle information represented by $[B_L, B_R]$ can be pruned up to $[A'_L, A'_R]$ (see Figure 10(b)) as the extremities of $[A_L, A_R]$ are closer to the obstacle boundaries than those of $[B_L, B_R]$.

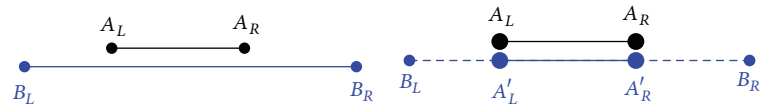
For the sake of clarity, obstacle information carried by vertical lines is ignored. Each collected obstacle information $[A_L, A_R]$ is iteratively compared to the remaining obstacle information. Let $[B_L, B_R]$ be the current obstacle information with which we compare $[A_L, A_R]$. A_L and B_L denote the left extremities of these two obstacle pieces of information, while A_R and B_R denote their right extremities. If $[A_L, A_R]$ and $[B_L, B_R]$ have similar directions, we try to prune their left extremities (resp., right extremities) with each other. Let us consider the left extremities of $[A_L, A_R]$ and $[B_L, B_R]$ (A_L and B_L). Firstly, we test if we can prune $[B_L, B_R]$ using A_L. To do so, the orthogonal projection A'_L of A_L on the line carrying $[B_L, B_R]$ has to be inside $[B_L, B_R]$ (see Figure 10(b)) and the Euclidean distance between A_L and A'_L has to be less than *MaxToleratedFarness*. $[B_L, B_R]$ can pruned using A_L if this condition is satisfied. So, the left extremity of $[B_L, B_R]$ (B_L) is pruned up to A'_L (B_L is replaced by A'_L). If $[B_L, B_R]$ could not be pruned using A_L, then we test if $[A_L, A_R]$ could be pruned using B_L (using the same previous procedure applied to $[B_L, B_R]$ and A_L). Afterwards, the same pruning procedure that we applied to $[A_L, A_R]$ and $[B_L, B_R]$ left extremities is performed for their right extremities. Algorithm 1 describes the pseudocode of the pruning step.

Let us investigate the pruning procedure execution time. Let M be the number of obstacle pieces of information in the input list L. Let t be the number of times the ProcessTC



(a) Before pruning (b) After pruning

Figure 9: A case of complementary obstacle information.



(a) Before pruning (b) After pruning

Figure 10: Another case of complementary obstacle information.

algorithm is called during the execution which is bounded by a constant according to the routing process. Then

$$M = \Theta\left(t * N^2\right). \tag{4}$$

Since we need to investigate each couple of obstacle pieces of information in the list L, the overall execution time of the pruning procedure is then

$$T = \Theta\left(M^2\right) = \Theta\left(N^4\right). \tag{5}$$

4.5. Step 3: Filtering Procedure. After the execution of the second step of the proposed obstacle detection procedure, the accuracy of obstacle information is significantly improved. Nevertheless, a given number of inaccurate or erroneous obstacle pieces of information may persist either due to the absence of pruning opportunity or to an erroneous obstacle detection. Subsequently, we have to conceive a dedicated filtering procedure permitting removing these inaccurate or erroneous obstacle pieces of information.

The proposed filtering procedure is based on the fact that inside and around the obstacle boundaries (see Figure 5(a)) the points representing the treated extremities of obstacle information are denser than elsewhere. As a result, we propose to eliminate a small percentage of obstacle information extremities within the least dense regions, namely, extremities beyond the contour of the obstacle. This percentage is controlled by a parameter denoted as *PercentageOfFilteredInformation* which is set by the user. As portrayed in Figure 5, such a simple heuristic succeeds in eliminating obstacle information extremities that are far from the real obstacle boundary.

The proposed filtering procedure works as follows. We start by calculating the local density of each point (obstacle information extremity). The local density of a given point P corresponds to the number of points (neighbours) whose distance to P is less than R. Then we calculate the local density histogram which will give useful statistical data based on which we select the filtering threshold (localDensityThreshold). Once we found the filtering threshold, we retain only the points having a local density greater than this filtering threshold. The complete pseudocode of the filtering procedure is described in Algorithm 2.

Let us calculate the filtering procedure execution time. Let M be the length of the list containing the pruned obstacle information. Recall that the length of this list is the same as that of list L of original collected obstacle pieces of information before pruning. In the beginning of the filtering procedure, the list of obstacle pieces of information is converted into a list of points (LP) which is performed in linear time as a function of M. Then, the LP list is used to calculate the local density of each point. This latter step has an execution time that is quadratic as a function of M. As MaxLocalDensity is bounded by the total number of points, then calculating the local density histogram is executed in linear time of M. Subsequently, the overall time complexity of the filtering procedure is

$$T = \Theta\left((M)^2\right). \tag{6}$$

4.6. Step 4: Concave Hull Construction. After the execution of the filtering step, we obtain a set of points (extremities of obstacle information) close to the real obstacle boundary. Then, we execute an algorithm to infer the contour of the detected obstacle by encompassing the retained points. In order to appropriately handle obstacles having small concavities, we have willingly chosen to use a concave hull construction algorithm rather than a convex hull algorithm. Recall that the proposed pruning scheme is primarily designed to handle only convex obstacles. But, as we will subsequently show, it can handle obstacles having small concavities. Contrary to convex hull, the construction of concave hull is not obvious. This is mainly due to the fact that for a given scatter plot there are usually a large number of possible concave hulls. In this paper, we have chosen to use a concave hull construction algorithm based on the one introduced in [27]. The main idea of this algorithm consists in building a concave hull starting from a convex hull obtained by any standard algorithm (in this paper we use the Jarvis March algorithm [28]). Then, for each edge of this hull, the algorithm decides whether it should dig inside the encountered concavity or not using a dedicated criterion called $N_{\text{threshold}}$ set by the user and

Require:
L: a list containing pairs of non communicating positions (obstacle information: OI).
MaxAngle: The maximum angle aperture tolerated between two given prunable OI.
MaxFarness: the maximum distance tolerated (i.e. to be prunable) between the projected OI extremity and its orthogonal projection on the line carrying the other OI.
Ensure: L: list of non communicating positions with more accurate extremities.

```
(1)  for (i = 1; i ≤ L.size; i = i + 1) do
(2)      for (j = i + 1; j ≤ L.size; j = j + 1) do
(3)          if (L[i].hasSimilarDirection(L[j], MaxAngle) = false) then
(4)              Continue                                  ▷ go to the next iteration
(5)          end if
(6)          Let D_i the line passing through L[i]
(7)          Let D_j the line passing through L[j]
(8)          Point p1 ← getOrthogonalProjection(L[i].LeftExtremity, D_j)
(9)          Point p2 ← getOrthogonalProjection(L[j].LeftExtremity, D_i)
(10)         d1 ← euclidianDistance(L[i].LeftExtremity, p1)
(11)         d2 ← euclidianDistance(L[j].LeftExtremity, p2)
(12)         Point p3 ← getOrthogonalProjection(L[i].RightExtremity, D_j)
(13)         Point p4 ← getOrthogonalProjection(L[j].RightExtremity, D_i)
(14)         d3 ← euclidianDistance(L[i].RightExtremity, p3)
(15)         d4 ← euclidianDistance(L[j].RightExtremity, p4)
(16)         if (p1.isInsideSegment(L[j]) and d1 ≤ MaxFarness) then
(17)             L[j].LeftExtremity ← p1;
(18)         else if (p2.isInsideSegment(L[i]) and d2 ≤ MaxFarness) then
(19)             L[i].LeftExtremity ← p2;
(20)         end if
(21)         if (p3.isInsideSegment(L[j]) and d3 ≤ MaxFarness) then
(22)             L[j].RightExtremity ← p3;
(23)         else if (p4.isInsideSegment(L[i]) and d4 ≤ MaxFarness) then
(24)             L[i].RightExtremity ← p4;
(25)         end if
(26)     end for
(27) end for
(28) return L
```

Algorithm 1: Pruning procedure.

which limits the nonsmoothness (of obstacle border) caused by the digging procedure.

The complete pseudocode of the concave hull construction step is detailed in Algorithm 3. In this pseudocode, we use a function called *FindNearestInnerPointToEdge* whose algorithm is described in Algorithm 4.

Now, we turn to calculate the execution time of the concave hull construction procedure. Let M be the number of the extremities of obstacle information retained after the filtering step. The calculation of the concave hull is initialized by a computation of the convex hull using the well-known Jarvis March algorithm. The complexity of the Jarvis March algorithm is equal to $\Theta(M*h)$ where h is the number of points (vertices) forming the convex hull [29]. In the worst case, the complexity of Jarvis March's algorithm is equal to $\Theta(M^2)$.

Then, the concave hull is iteratively refined using a digging criterion as follows. For each edge, we try to find the nearest inner point (P) to it. P is the nearest point of G to E such that none of the neighbouring edges of E is closer to P than E. If P exists, we test whether the digging criterion ((L/decisionDistance) > $N_{\text{threshold}}$) is satisfied or not. If verified, the current edge is exploded into two new edges (formed by the extremities of the current edge and the point P). In the worst case, we can dig up to M times into the concave hull.

On the other hand, the computational time of Algorithm 4 which permits to find the nearest inner point P (if it exists) is $\Theta(M)$. Subsequently, the overall execution time of the concave hull construction procedure is then

$$T = \Theta\left(M^2\right). \tag{7}$$

5. Results and Discussion

In this section, we start by describing the different parameters used in conducted simulations. Then, we define various evaluation metrics required to assess the performance of our obstacle detection scheme. After that, we overview the scheme used to select the best parameters values that maximize the performance of our obstacle detection scheme. Finally, we detail and then discuss the results of our simulation scenarios.

```
Require:
    PrunedList: a list of obstacle information (OI) resulting fom the pruning step.
    R: Local density radius.
    percentageOfObstInfToBeFiltered: percentage of OI to be filtered out.
Ensure: FilteredList: a list containing the kept points (positions).
            ▷ convert the list of position pairs (OI) to a list of positions (points)
(1)  LP: List                                                          ▷ list of points
(2)  for each obstacleInformation O in PrunedList do
(3)      LP.Add(O.LeftExtremity)
(4)      LP.Add(O.RightExtremity)
(5)  end for
          ▷ For each point we calculate its local density (i.e. the number of points (neighbours) whose distance
     with the current point is less than R
(6)  nbOfNeighbours[LP.size] ▷ array in which we save the local density of each point
(7)  for (i = 1; i ≤ LP.size; i = i + 1) do
(8)      nbOfNeighbours[i] ← 0
(9)  end for
(10) maxDensity ← 0
(11) for (i = 1; i ≤ LP.size; i = i + 1) do
(12)     for (j = 1; j ≤ LP.size; j = j + 1) do
(13)         if (euclidianDistance(LP[i], LP[j]) < R) then
(14)             nbOfNeighbours[i]++
(15)         end if
(16)     end for
(17)     if (nbOfNeighbours[i] > maxDensity) then
(18)         maxDensity ← nbOfNeighbours[i]
(19)     end if
(20) end for
(21) locDensHist[maxDensity + 1]      ▷ an array in which we will calculate the local density histogram
(22) for (i = 1; i ≤ locDensHist.size; i = i + 1) do
(23)     locDensHist[i] ← 0
(24) end for
(25) for (i = 1; i ≤ LP.size; i = i + 1) do
(26)     locDensHist[nbOfNeighbours[i]]++
(27) end for
(28) nbOfObstacleInformationToBeFiltered ← LP.size/100 ∗ percentageOfObstInfToBeFiltered
(29) sumOfObstInf ← 0
(30) localDensityThreshold ← positiveInfinity
(31) for (i = 1; i ≤ locDensHist.size; i = i + 1) do
(32)     sumOfObstInf += localDensityHistogram[i]
(33)     if (sumOfObstInf > nbOfObstacleInformationToBeFiltered) then
(34)         localDensityThreshold ← i
(35)         break                                                     ▷ leave the loop
(36)     end if
(37) end for
(38) FilteredList: List
(39) for (i = 1; i ≤ LP.size; i = i + 1) do
(40)     if (nbOfNeighbours[i] ≥ localDensityThreshold) then
(41)         FilteredList.Add(LP[i])
(42)     end if
(43) end for
(45) return FilteredList
```

Algorithm 2: Filtering procedure.

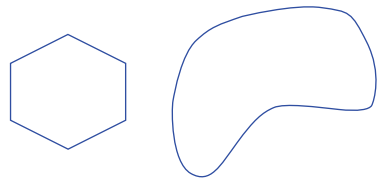
5.1. Simulation Setup. We develop our proposed obstacle detection scheme under INETMANET Framework within the OMNeT++ network simulator (Version 4.1). The simulated MANET area is equal to 500 m by 500 m. In this network, 60 mobile nodes are initially scattered. These nodes follow the Random Way Point mobility model [30] with a null wait time, an update interval of 0.1 s, and a constant speed. The node transmission range is fixed to 200 m. The network capacity is set to 54 Mbps. CE-OLSR protocol parameters are set as follows. TC_redundancy is set to 2, which means that

Require:
G: list of points retained after the filtering step.
$N_{\text{threshold}}$: the threshold based on which we decided to dig or not into the encountered concavities.
Ensure: ConcaveHullEdges: a List containing the edges forming the built concave hull of G.
(1) ConvexHullPoints ← **BuildConvexHull**(G) ▷ build the points list forming the convex hull that encompasses G scatterplot using the well known Jarvis March algorithm
(2) ConvexHullEdges: list
(3) **for** (i = 1; $i \leq$ ConvexHullPoints.size; $i = i + 1$) **do** ▷ convert ConvexHullPoints to a list of edges
(4) $j = (i + 1)$ mod ConvexHullPoints.size
(5) Point $c1$ ← envC[i]
(6) Point $c2$ ← envC[j]
(7) Edge e ($c1, c2$)
(8) ConvexHullEdges.**pushBack**(e)
(9) **end for**
(10) ConcaveHullEdges ← ConvexHullEdges
(11) $G \leftarrow G$ − ConvexHullPoints
(12) **for each** edge E **in** ConvexHullEdges **do**
(13) P ← **FindNearestInnerPointToEdge**(G, E)
▷ P is the nearest point of G to E such as neither of neighbour edges of E is closer to P than E
(14) **if** (P exists) **then**
(15) L ← length of the edge E ▷ L = **euclidianDistance**(E.ext1, E.ext2)
(16) decisionDistance ← **distanceToEdge**(P, E)
(17) **if** ((L/decisionDistance) > $N_{\text{threshold}}$) **then**
(18) insert new edges E_1(E.ext1, P) and E_2(E.ext2, P) into the tail of ConcaveHullEdges
(19) delete the edge E from the ConcaveHullEdges
(20) $G \leftarrow G - P$ ▷ delete P from G
(21) **end if**
(22) **end if**
(23) **end for**
(24) **Return** ConcaveHullEdges

Algorithm 3: Concave hull construction step.

Require:
G: List of points.
E: edge for which we try to find the nearest inner point.
Ensure: Nearest inner point to E.
(1) **function** FindNearestInnerPointToEdge(G, E)
(2) minDistance ← positiveInfinity
(3) nearestPoint ← Null
(4) **for each** point P **in** G **do**
(5) previousEdge ← E.previousEdge
(6) nextEdge ← E.nextEdge
(7) distToPreviousEdge ← **distanceToEdge**(previousEdge, P)
(8) distToNextEdge ← **distanceToEdge**(nextEdge, P)
(9) distToE ← **distanceToEdge**(E, P)
(10) **if** ((distToE < minDistance) **and** (distToE < distToPreviousEdge) **and** (distToE < distToNextEdge)) **then**
(11) minDistance ← distToE
(12) nearestPoint ← P
(13) **end if**
(14) **end for**
(15) **return** nearestPoint
(16) **end function**

Algorithm 4: FindNearestInnerPointToEdge function.



(a) Convex obstacle (b) Concave obstacle

Figure 11: Two MANETs with an obstacle in the middle.

MPR nodes publish the entire list of their neighbours in their TC messages. The TC message periodicity is set to 8 s and that of Hello messages is equal to 2 s. The OLSR sending jitter is randomly picked from the [0, 0.5] interval. The CE-OLSR stability distance parameter is fixed to 50 m.

The performance of obstacle detection is assessed using dedicated metrics (precision ratio and coverage ratio) that we detail in the following section. The unique parameter which is not subject to parameter optimization is the reduced range parameter. Recall that this reduced range is used to build the L_2 list in the first step of our obstacle detection scheme. Omitting this parameter from optimization procedure returns to the fact that it is mainly related to nodes' speed. We willingly set the reduced range value to 150 m; simulation results show that such a value is suitable for node speeds ranging up to 10 m/s. Further details about parameter selection and optimization will be later detailed in Section 5.2.4.

During the conducted simulations, each node runs the obstacle detection procedure every 50 s in order to assess the time effect on our scheme. But, in practice, the periodicity of obstacle detection could be much more relaxed to preserve network resources. Finally, we consider a network containing only one static obstacle (convex or concave) as portrayed in Figures 11(a) and 11(b). For each obstacle type (convex/concave), we consider 3 different sizes that we name a large, a medium, and a small obstacle. For the convex obstacle case, the bounding boxes of the 3 considered sizes are as follows: 120 m by 120 m, 60 m by 60 m, and 30 m by 30 m. For the concave obstacle case, we consider also 3 sizes whose bounding boxes are as follows: 216 m by 180 m, 108 m by 90 m, and 54 m by 45 m.

5.2. Evaluation Metrics. The ultimate goal of our obstacle detection scheme consists in inferring the obstacle boundaries solely using CE-OLSR signalling. To the best of our knowledge, there is no other research work that attempted to infer the obstructing obstacle boundaries using basic routing messages or another usual signalling of MANETs. For that reason we self assess our obstacle detection scheme using dedicated metrics. The first metric called *coverage ratio* reflects the completeness of the obstacle detection process. The second metric named *precision ratio* reveals the closeness of the obstacle detection output to the real obstacle boundaries. We also use the *route validity* metric to assess the impact of our obstacle detection strategy on the overall performance of CE-OLSR.

5.2.1. Coverage Ratio. The coverage ratio metric highlights the completeness of the obstacle detection. It measures the ability of our obstacle detection approach to cover the real obstacle area. According to (8), the coverage ratio is defined as the ratio of the correctly detected obstacle area (CDOA) to the real obstacle area (ROA):

$$\text{Coverage Ratio} = \frac{\text{CDOA}}{\text{ROA}}. \tag{8}$$

Figure 12 schematically depicts the different areas used in our proposed metrics. In the left part of Figure 12(a), we portray a case of nonoptimal parameters values (MaxToleratedFarness = 40 m, MaxAngleAperture = 10, LocalDensityRadius = 6 m, $N_{\text{threshold}}$ = 6, and PercentageOfFilteredInformation = 6%) in order to highlight the different regions accounted in our metrics. In this figure, ROA relates to the network area in which the real obstacle resides. The CDOA corresponds to the common area (i.e., the intersection) between the total area detected as obstacle (TADO) and the real obstacle area (ROA). When the real obstacle area is totally covered by the TADO, the coverage ratio is equal to 1.

5.2.2. Precision Ratio. The precision ratio quantifies the precision of an obstacle detection strategy. As it is shown in (9), the precision ratio is calculated by subtracting the detection error from 1. The detection error is quantified by dividing the sum of the detected area outside the obstacle (DAOO) and the nondetected area of the real obstacle (NDA) by the union of the areas of TADO and ROA. The precision ratio ranges from 0 to 1. Values close to 0 reflect a weak obstacle detection, while values close to 1 mean a good obstacle detection:

$$\text{Precision Ratio} = 1 - \frac{\text{DAOO} + \text{NDA}}{\cup(\text{TADO}, \text{ROA})}. \tag{9}$$

5.2.3. Routes Validity. In addition to the aforementioned metrics (precision and coverage ratio), we define a third metric that measures the impact of our obstacle detection scheme on the performance of CE-OLSR routing decisions. The performance of routing decisions is assessed using the routes validity metric which measures the consistency of the routing table of a given node compared to the real network topology. Note that, in both OLSR and CE-OLSR, each routing table entry points only to the next gateway that leads to the ultimate destination. So, to know the whole route toward a given destination, we have to warp through the routing tables of intermediate gateways until reaching the destination. A given route is termed valid if it does exist in the real network topology (as maintained by the simulator). The routes that are marked as unreachable in the routing table of a given node are also termed valid if they do not exist in the real network topology. In the remaining cases, the route is considered invalid. The validity of the routes of a given node is defined as the percentage of valid routes retained in its routing table.

5.2.4. Selection of Obstacle Detection Parameters. In our obstacle detection scheme, we use several parameters that

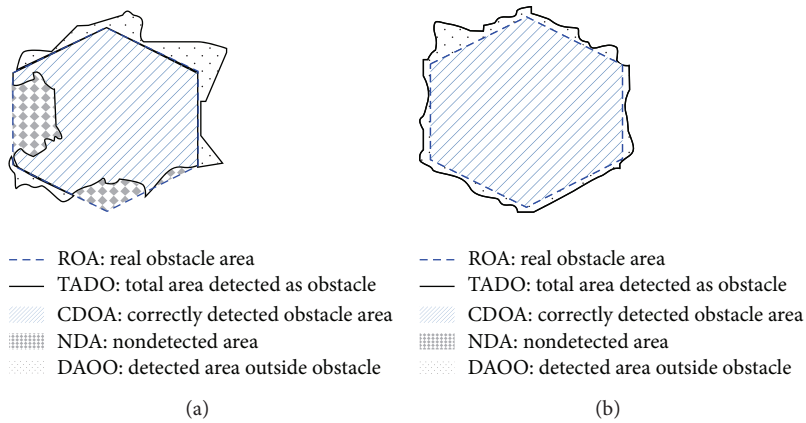


Figure 12: Different areas considered in evaluation metrics (large hexagone, speed = 5 m/s): (a) simulation time = 100 s, MaxToleratedFarness = 40 m, MaxAngleAperture = 10, LocalDensityRadius = 6 m, $N_{\text{threshold}} = 6$, and PercentageOfFilteredInformation = 6%; (b) simulation time = 1000 s, MaxToleratedFarness = 5 m, MaxAngleAperture = 5, LocalDensityRadius = 9 m, $N_{\text{threshold}} = 6$, and PercentageOfFilteredInformation = 6%.

Require: L: a list containing pairs of non communicating positions.
Ensure: List of obstacle detection parameters values leading to the best performance.
(1) CN ← 0 ▷ CN: CombinationNumber
(2) T ← 800 ▷ T: Time (seconds)
(3) **for each** PercentageOfFilteredInformation (POI) **from** 2 **to** 6 **step** 2 **do**
(4) **for each** MaxAngleAperture (Ang) **from** 5 **to** 20 **step** 5 **do**
(5) **for each** LocalDensityRadius (R) **from** 3 **to** 9 **step** 3 **do**
(6) **for each** MaxToleratedFarness (F) **from** 5 **to** 45 **step** 5 **do**
(7) **for each** $N_{\text{threshold}}$ **from** 4 **to** 10 **step** 2 **do**
(8) CN ← CN + 1
(9) **for each** run **from** 1 **to** 35 **step** 1 **do**
(10) detectedObstcle ← **detectObstcle**(POI, Ang, R, F, $N_{\text{threshold}}$, T)
(11) Precision[run] ← **calculatePrecision**(detectedObstcle, realObstcle)
(12) Coverage[run] ← **calculateCoverage**(detectedObstcle, realObstcle)
(13) **end for**
(14) AveragePrecision ← **average**(Precision{1, ⋯ , 35})
(15) AverageCoverage ← **average**(Coverage{1, ⋯ , 35})
(16) optimizationCriteria[CN] ← (AveragePrecision + AverageCoverage)/2
(17) **end for**
(18) **end for**
(19) **end for**
(20) **end for**
(21) **end for**
bestPerformanceCombination ← **indexOf**(**maximum** (optimizationCriteria{1, . . . , 1296}))
(22) **return** Parameters corresponding to bestPerformanceCombination

Algorithm 5: Selection procedure.

control the operation of the pruning, the filtering, and the concave hull construction procedures. For each parameter, we have to find the value which optimizes the performance of our obstacle detection scheme.

To perform this selection/optimization procedure, we need a ground truth on which we measure the quality of obstacle detection. In this work, we use the conceived metrics (precision/coverage ratios) as a ground truth on which we perform parameter selection procedure. We iteratively test several combinations of all parameters values. Then we retain the combination that leads to the best performance in terms of the average of precision and coverage ratio. Overall, we tested 1296 combinations for each obstacle size, each node speed, and each obstacle type (convex/concave).

The pseudocode of the parameter selection procedure is depicted in Algorithm 5. Table 1 summarises the values of the different parameters that lead to the best performance in every case considered in this paper. We note that the following parameters combination operates well in several obstacle sizes and nodes speeds: "MaxToleratedFarness" = 45 m,

Table 1: Obstacle detection parameters leading to the best performance.

Obstacle type	Obstacle size	Speed	R	MaxAngleAperture	MaxToleratedFarness	PercentageOfFilteredInformation	$N_{\text{threshold}}$
Hexagone	Small	5	6	5	45	6	8
Hexagone	Medium	2	6	5	45	6	8
Hexagone	Medium	5	6	5	45	6	6
Hexagone	Medium	10	6	5	45	6	6
Hexagone	Large	5	9	5	5	6	6
Concave	Medium	5	9	5	5	2	8

"MaxAngleAperture" = 5 degrees, "LocalDensityRadius" (R) = 6 m, "PercentageOfFilteredInformation" = 6%, and "$N_{\text{threshold}}$" = 8.

5.3. Simulation Results. Our simulation scenarios target the following:

(i) investigating the ability of our obstacle detection scheme to detect either convex obstacles or obstacles having slight concavities;

(ii) studying the impact of nodes velocity as well as obstacle size on the accuracy of the obstacle detection;

(iii) assessing the impact of our obstacle detection scheme on the routes validity of CE-OLSR;

(iv) evaluating the storage complexity of our proposed scheme.

5.3.1. Obstacle Boundaries Detection Capabilities. In the first simulation set, we assess the ability of our obstacle detection scheme to detect convex obstacles and obstacles having slight concavities. Recall that our obstacle detection scheme is not conceived in the first place to handle concave obstacles. However, we will show that our scheme is able to cope with obstacles having slight concavities.

Figure 13 (resp., Figure 14) portrays the evolution of the coverage and precision ratios over time in a MANET that contains a single convex (resp., concave) obstacle having a medium size which is situated in the center of the network. The node speed here is set to 5 m/s. Obtained results show that both of the coverage ratio and the precision ratio metrics progressively increase over time. But the coverage ratio metric converges faster than the precision ratio.

According to Figures 13 and 14, for both obstacle types (convex/concave), we reach a coverage ratio greater than or equal to 0.98 in just 50 seconds. Note that, for a convex obstacle, the coverage ratio slightly drops (beyond 50 s) to 0.95. This slight decrease is coupled with a noticeable increase of the precision ratio which reaches 0.8 in just 200 s. We also notice that the precision ratio is better and it converges faster in the case of a convex obstacle. Indeed, as depicted in Figures 13 and 14, the stationary regime of both metrics is reached in just 400 s in the case of convex obstacle (precision ratio = 0.85). But in the case of concave obstacle, it is required to run up to 900 s to reach this stationary regime (precision ratio = 0.71). Figure 15 shows the result of our single obstacle

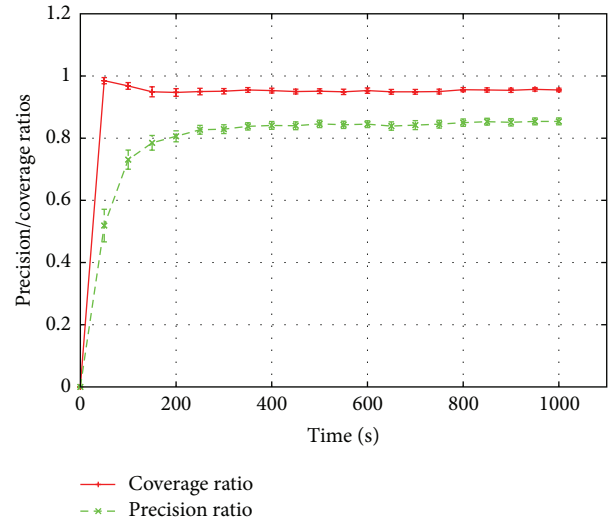


Figure 13: Coverage and precision ratios: medium convex obstacle, speed = 5 m/s.

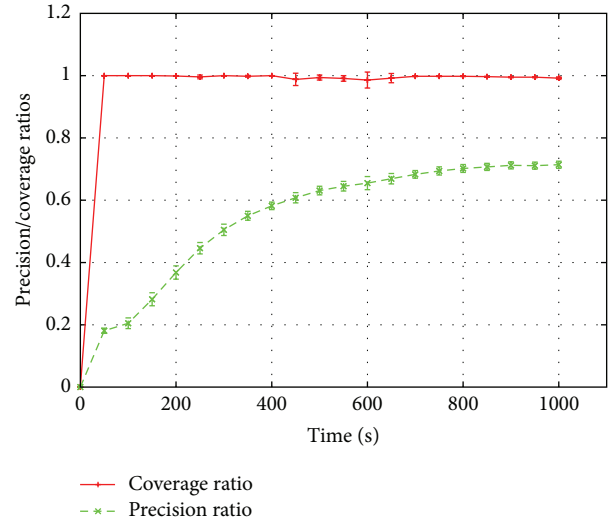


Figure 14: Coverage and precision ratios: medium concave obstacle, speed = 5 m/s.

detection scheme obtained for a simulation time equal to 1000 seconds.

Notice that our obstacle detection scheme provides an excellent coverage ratio with an adequate precision ratio.

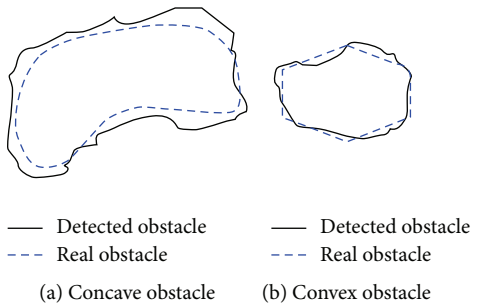


Figure 15: Obstacle detection result: nodes speed = 5 m/s, simulation time = 1000 s, and obstacle size = medium.

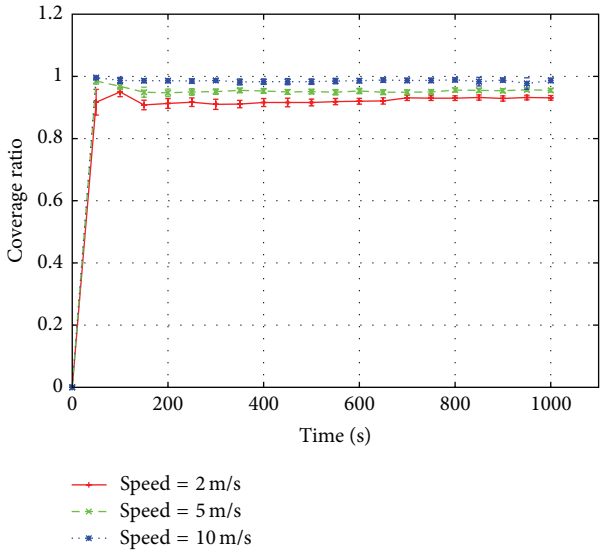


Figure 16: Speed effect on coverage ratio: medium convex obstacle.

Recall that our objective is to avoid the hard assumption considered in our previously conceived CE-OLSR protocol with obstacle awareness [6]. In such a protocol, every node is supposed to be aware of the map of obstacles that reside in the network area. Our proposed obstacle detection scheme shows a slight overestimation of the obstacle area but this is not critical and could only further shield the obstacle. On the contrary, underestimating the obstacle area is critical as the routing protocol may select links that cross the obstructing obstacle.

5.3.2. Node Velocity Impact. We now investigate the impact of the nodes speed on the precision and coverage ratio metrics. According to Figure 16, the coverage ratio metric is affected by the decrease in nodes speed. Such behaviour could be explained by the fact that for low speeds the probability to detect noncommunicating positions that are closer to the obstacle boundary is higher. So, after running the concave hull construction step, the risk of digging into the real obstacle area is higher than in the case of high speeds. In addition, the periodicity of Hello messages which is set to 2 s participates in delaying the perception of the nodes locations

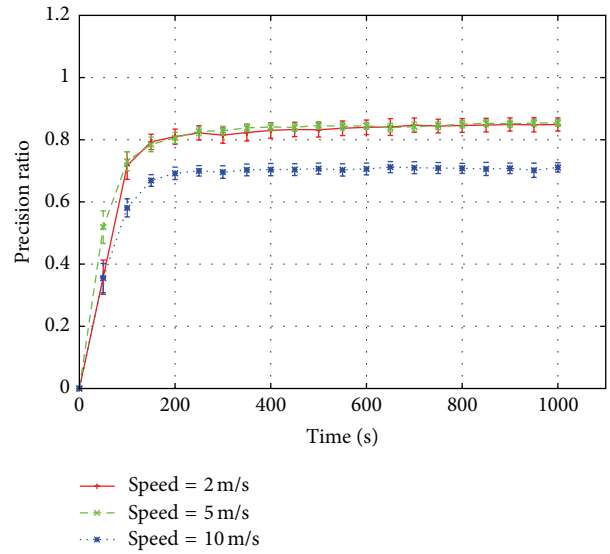


Figure 17: Speed effect on precision ratio: medium convex obstacle.

changes. Subsequently, the higher the speed is, the more our obstacle detection scheme is subject to overestimating the obstacle area. This observation is confirmed by the results obtained in Figure 17. Indeed, the precision ratio metric degrades with the increase in nodes speed. For a low speed of 2 m/s, the precision ratio metric increases progressively over time up to reaching 0.85 within 1000 s simulation time. But for a high speed equal to 10 m/s the precision ratio hardly reaches 0.71 for the same simulation time (1000 s). For a medium speed of 5 m/s, we obtain an average performance in terms of the precision ratio similar to that of of 2 m/s.

5.3.3. Obstacle Size Impact. In the third simulation set, we study the impact of obstacle size on the considered evaluation metrics. Figure 18 shows that our obstacle detection scheme is able to reach a coverage ratio greater than 0.95 in just 50 s, for all considered convex obstacle sizes. For large convex obstacle, we obtain a high coverage (>0.99) ratio during almost the whole simulation time that follows the convergence time. Figure 19 shows that obstacle size impacts the precision ratio metric. Indeed, for large and medium obstacle sizes, we obtain a precision ratio greater than 0.83. But for small obstacles the precision is just about 0.74. In addition, the convergence time of the precision ratio metric in the case of small and medium obstacles is better than that of a large obstacle. For a small and medium obstacle, we reach the steady state in about 400 s but for a large obstacle the precision continues increasing up to the end of the simulation time.

5.3.4. Impact on Route Validity. In the fourth simulation set, we assess the after effect of our obstacle detection scheme on the routes validity of CE-OLSR. Figure 20 portrays the performance of CE-OLSR with obstacle detection against the performance of CE-OLSR with and without obstacle awareness. In these simulations, obstacle detection is performed by

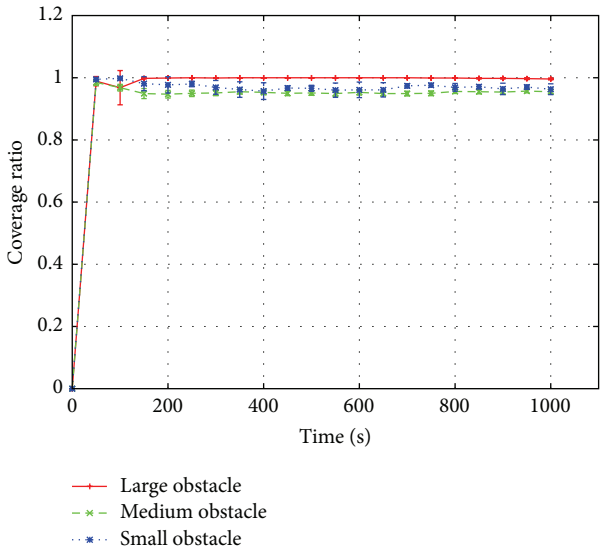


Figure 18: Effect of obstacle size on coverage ratio: convex obstacle, speed = 5 m/s.

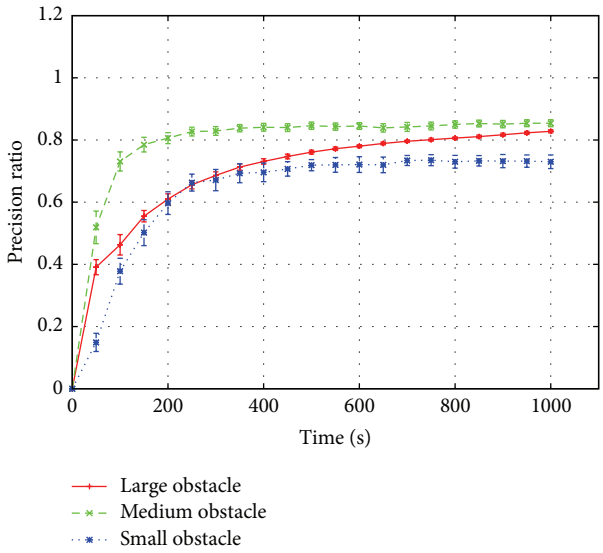


Figure 19: Effect of obstacle size on precision ratio: convex obstacle, speed = 5 m/s.

each node every 20 s. We willingly reduced the periodicity of obstacle detection to see in a timely manner the impact of obstacle detection on the routes validity. As we mentioned earlier, in practice, obstacle detection period could be increased to preserve nodes resources. Note that each point of Figure 20 denotes the average validity of the routes calculated by all nodes of the MANET at the corresponding simulation instants. As we can see in Figure 20, the routes validity of CE-OLSR without awareness about the obstacle map is highly affected compared to CE-OLSR with obstacle awareness. In CE-OLSR with obstacle awareness, the routes validity ranges between 96% and 100% almost all the time. In CE-OLSR without prior knowledge of the obstacle map,

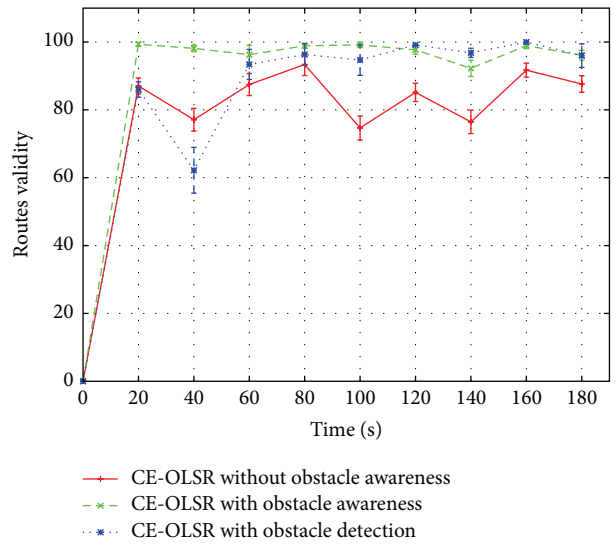


Figure 20: After effect of obstacle detection on the routability of CE-OLSR: medium convex obstacle, speed = 5 m/s.

routes validity drops down to 74% many times during the simulation. When we apply our obstacle detection scheme, the validity of the routes achieved by CE-OLSR becomes similar to that of CE-OLSR with obstacle awareness. Notice that for a simulation time less than 60 s mobile nodes do not have sufficient obstacle information to perform obstacle detection. In some curve plots (time = 140 s, 160 s, and 180 s), the routes validity achieved by our newly conceived protocol (CE-OLSR with obstacle detection) is even better than that of CE-OLSR with obstacle awareness. This is essentially due to the slight overestimation of the obstacle area which avoids using weak links (i.e., links that pass near the obstacle area). Indeed, such weak links may be broken rapidly due to nodes mobility.

5.3.5. Storage Complexity Impact. Now we turn to investigate the storage complexity of our proposed obstacle detection scheme. Even though our obstacle detection scheme does not add any signalling overhead to the underlying protocol, its operation requires some additional storage capabilities in every node participating in the MANET. In fact, at the reception of a TC message, each node may infer new obstacle information that it has to be stored locally for a later usage. The list of inferred obstacle pieces of information grows over time. Each piece of information is composed of a pair of node positions. As such, each obstacle piece of information requires 4 times the size required to store an integer. Note that the position of nodes is by default double in OMNET++. But, for our usage, we keep only the integer part. Figures 21 and 22 portray the effect of speed (resp., obstacle size) on the number of collected obstacle pieces of information. Figure 21 shows that for a low speed of 2 m/s nodes collect more obstacle information than at higher speeds. This is an expected result because, for such a low speed, nodes spend more time in the vicinity of the obstacle. Subsequently, it is more likely to discover new obstacle information than

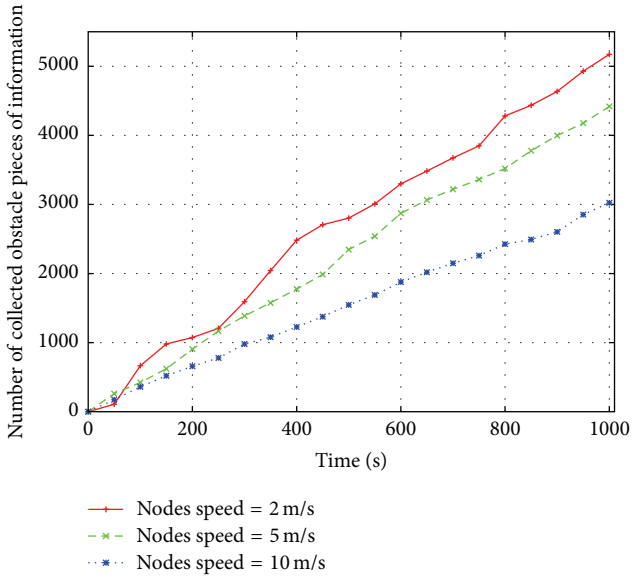


Figure 21: Nodes speed effect on the number of collected obstacle pieces of information: medium convex obstacle.

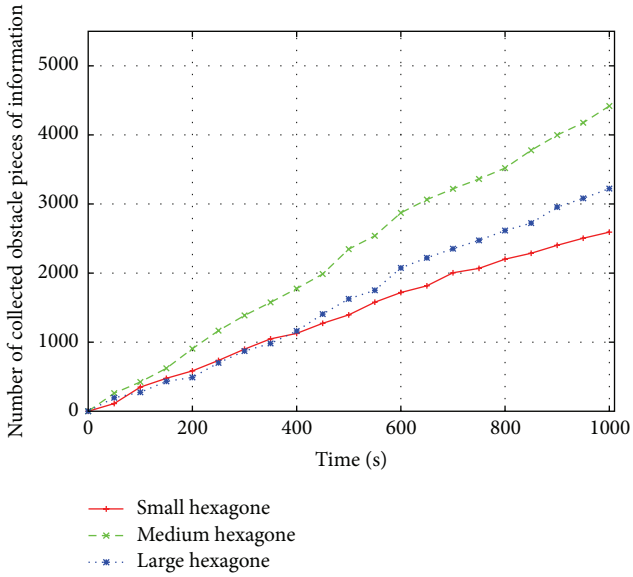


Figure 22: Obstacle size effect on the number of collected obstacle pieces of information: medium convex obstacle, nodes speed = 5 m/s.

higher speeds. Figure 22 shows that for a medium obstacle nodes collect more obstacle information than the case of large or small convex obstacle. This behaviour is essentially due to the transmission range (150 m) which is close to the obstacle diameter (bounding box = 120 m × 120 m). It follows that the likelihood of detection of new obstacle information decreases, whereas in the case of a small obstacle the decrease of obstacle size lowers the likelihood of finding noncommunicating positions pairs.

6. Conclusions

The signalling of CE-OLSR, which is the same as that of OLSR, is used to automatically and dynamically infer the contour of static obstacles standing within the network. We developed an autonomous obstacle detection scheme to delimit static convex obstacles. Simulation results showed that such a scheme is also capable of delimiting obstacles with slight concavities.

We defined dedicated metrics, namely, the coverage ratio and the precision ratio, respectively, measuring the ability of our obstacle detection scheme to cover the real obstacle area and the precision of such a detection. Obtained results showed the effectiveness of our proposal as it nicely covers the entire obstacle area for all considered sizes and types (convex/concave) of obstacles regardless of node speeds. The proposed detection scheme provides an adequate precision ratio sufficient to efficiently fulfill our objective of autonomously avoiding broken links caused by the obstructing obstacle without *"a priori"* knowledge of the obstacle map.

Our proposed detection scheme is then integrated into CE-OLSR to allow the automatic detection and avoidance of obstacles that might exist in the network. Simulation results showed that CE-OLSR augmented with our proposed detection scheme provides the exact same route validity as the CE-OLSR-OA which has the *a priori* knowledge of the obstacle map.

Further investigations are underway for detecting multiple mobile obstacles and general concave obstacle contours.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

The authors extend their appreciation to the Deanship of Scientific Research at King Saud University for funding this work through research group no. RGP-1436-002.

References

[1] M. Conti and S. Giordano, *Multihop Ad Hoc Networking: The Evolutionary Path*, John Wiley & Sons, 2013.

[2] M. A. Abid and A. Belghith, "Period size self tuning to enhance routing in MANETs," *International Journal of Business Data Communications and Networking*, vol. 6, no. 4, pp. 21–37, 2010.

[3] M. Conti and S. Giordano, "Mobile ad hoc networking: milestones, challenges, and new research directions," *IEEE Communications Magazine*, vol. 52, no. 1, pp. 85–96, 2014.

[4] A. Belghith and M. A. Abid, "Autonomic self tunable proactive routing in mobile ad hoc networks," in *Proceedings of the 5th IEEE International Conference on Wireless and Mobile Computing Networking and Communication (WiMob '09)*, pp. 276–281, October 2009.

[5] A. Goldsmith, M. Effros, R. Koetter, M. Médard, A. Ozdaglar, and L. Zheng, "Beyond Shannon: the quest for fundamental performance limits of wireless ad hoc networks," *IEEE Communications Magazine*, vol. 49, no. 5, pp. 195–205, 2011.

[6] A. Belghith and M. Belhassen, "CE-OLSR: a cartography and stability enhanced OLSR for dynamic MANETs with obstacles," *KSII Transactions on Internet and Information Systems*, vol. 6, no. 1, pp. 290–306, 2012.

[7] Y. Xiang, Z. Liu, R. Liu, W. Sun, and W. Wang, "Geosvr: a map-based stateless vanet routing," *Ad Hoc Networks*, vol. 11, no. 7, pp. 2125–2135, 2013.

[8] M. Belhassen, A. Belghith, and M. A. Abid, "Performance evaluation of a cartography enhanced OLSR for mobile multi-hop ad hoc networks," in *Proceedings of the Wireless Advanced (WiAd '11)*, pp. 149–155, IEEE, London, UK, June 2011.

[9] W.-T. Wang and K.-F. Ssu, "Obstacle detection and estimation in wireless sensor networks," *Computer Networks*, vol. 57, no. 4, pp. 858–868, 2013.

[10] L. Moraru, P. Leone, S. Nikoletseas, and J. D. P. Rolim, "Near optimal geographic routing with obstacle avoidance in wireless sensor networks by fast-converging trust-based algorithms," in *Proceedings of the 3rd ACM Workshop on QoS and Security for Wireless and Mobile Networks (Q2SWinet '07)*, pp. 31–38, ACM, October 2007.

[11] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," in *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking (MobiCom '03)*, pp. 134–146, ACM, New York, NY, USA, September 2003.

[12] B. Karp and H. T. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in *Proceedings of the 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '00)*, pp. 243–254, Boston, Mass, USA, August 2000.

[13] P. Bose, P. Morin, I. Stojmenović, and J. Urrutia, "Routing with guaranteed delivery in ad hoc wireless networks," *Wireless Networks*, vol. 7, no. 6, pp. 609–616, 2001.

[14] L. Moraru, P. Leone, S. Nikoletseas, and J. Rolim, "Geographic routing with early obstacles detection and avoidance in dense wireless sensor networks," in *Proceedings of the 7th International Conference on Ad-Hoc, Mobile and Wireless Networks (ADHOC-NOW '08)*, pp. 148–161, Springer, Sophia-Antipolis, France, September 2008.

[15] A. Koutsopoulos, S. Nikoletseas, and J. D. P. Rolim, "Near-optimal data propagation by efficiently advertising obstacle boundaries," in *Proceedings of the 6th ACM International Symposium on Performance Evaluation of Wireless Ad-Hoc, Sensor, and Ubiquitous Networks (PE-WASUN '09)*, pp. 15–22, ACM, Canary Islands, Spain, October 2009.

[16] F. Huc, A. Jarry, P. Leone, L. Moraru, S. Nikoletseas, and J. Rolim, "Early obstacle detection and avoidance for all to all traffic pattern in wireless sensor networks," in *Algorithmic Aspects of Wireless Sensor Networks*, S. Dolev, Ed., vol. 5804 of *Lecture Notes in Computer Science*, pp. 102–115, Springer, Berlin, Germany, 2009.

[17] F. Huc, A. Jarry, P. Leone, and J. Rolim, "Brief announcement: routing with obstacle avoidance mechanism with constant approximation ratio," in *Proceedings of the 29th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC '10)*, pp. 116–117, ACM, New York, NY, USA, July 2010.

[18] B. C. Seet, G. Liu, B.-S. Lee, C.-H. Foh, K.-J. Wong, and K.-K. Lee, "A-STAR: a mobile ad hoc routing strategy for metropolis vehicular communications," in *Proceedings of the 3rd International IFIP-TC6 Networking Conference (Networking '04)*, pp. 989–999, Athens, Greece, May 2004.

[19] C. Lochert, H. Hartenstein, J. Tian, H. Fübler, D. Hermann, and M. Mauve, "A routing strategy for vehicular ad hoc networks in city environments," in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV '03)*, pp. 156–161, June 2003.

[20] C. Lochert, M. Mauve, H. Füßler, and H. Hartenstein, "Geographic routing in city scenarios," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 9, no. 1, pp. 69–72, 2005.

[21] T. Clausen and P. Jacquet, "Optimized link state routing protocol (olsr)," Experimental RFC 3626, 2003.

[22] C. Adjih and L. Viennot, "Computing connected dominated sets with multipoint relays," *Journal of Ad Hoc and Sensor Wireless Networks*, vol. 1, pp. 27–39, 2002.

[23] A. Qayyum, L. Viennot, and A. Laouiti, "Multipoint relaying for flooding broadcast messages in mobile wireless networks," in *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS '02)*, vol. 9, pp. 3866–3875, IEEE Computer Society, Washington, DC, USA, January 2002.

[24] Y. Ge, T. Kunz, and L. Lamont, "Quality of service routing in ad-hoc networks using olsr," in *Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS '03)*, p. 300, 2003.

[25] H. Badis and K. A. Agha, "QOLSR, QoS routing for ad hoc wireless networks using OLSR," *European Transactions on Telecommunications*, vol. 16, no. 5, pp. 427–442, 2005.

[26] M. A. Abid and A. Belghith, "Stability routing with constrained path length for improved routability in dynamic manets," *Personal and Ubiquitous Computing*, vol. 15, no. 8, pp. 799–810, 2011.

[27] J.-S. Park and S.-J. Oh, "A new concave hull algorithm and concaveness measure for n-dimensional datasets," *Journal of Information Science and Engineering*, vol. 28, no. 3, pp. 587–600, 2012.

[28] R. A. Jarvis, "On the identification of the convex hull of a finite set of points in the plane," *Information Processing Letters*, vol. 2, no. 1, pp. 18–21, 1973.

[29] M. M. Atwah, J. W. Baker, S. Akl, and J. W. B. S. Akl, "An associative implementation of classical convex hull algorithms," in *Proceedings of the 8th IASTED International Conference on Parallel and Distributed Computing and Systems*, pp. 435–438, 1996.

[30] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Communications and Mobile Computing*, vol. 2, no. 5, pp. 483–502, 2002.