A Model for Obstacles to be used in Simulations of Wireless Sensor Networks and its Application in studying Routing Protocol Performance

Ioannis Chatzigiannakis, Georgios Mylonas and Sotiris Nikoletseas
Computer Technology Institute (CTI) and
Department of Computer Engineering and Informatics,
University of Patras, 26500 Patras, Greece.
{ichatz, mylonasg, nikole}@cti.gr
June 14, 2007

Abstract

In this work, we propose a simple obstacle model to be used while simulating wireless sensor networks. To the best of our knowledge, this is the first time such an integrated and systematic obstacle model for these networks appears. We define several types of obstacles that can be found inside the deployment area of a wireless sensor network and provide a categorization of these obstacles, based on their nature (physical and communication obstacles, i.e., obstacles that are formed out of node distribution patterns or have physical presence, respectively), their shape, as well as their change of nature over time. We make an extension to a custom-made sensor network simulator (simDust) and conduct a number of simulations in order to study the effect of obstacles on the performance of some representative (in terms of their logic) data propagation protocols for wireless sensor networks. Our findings confirm that obstacle presence has a significant impact on protocol performance, and also that different obstacle shapes and sizes may affect each protocol in different ways, thus giving insight to how will a routing protocol perform in the case of obstacles’ existence and indicate possible protocol shortcomings. Moreover, our results show that the effect of obstacles is not directly related to the density of a sensor network, and cannot be emulated only by changing the network density.

Keywords: Applications in Computer Networks, Sensor Networks, Modeling and Simulation Environments, Parallel and Distributed Computing, Performance evaluation, Obstacle avoidance, Routing.

1 Introduction

Wireless sensor networks are large collections of small in size, low-power, low-cost sensor devices that collect and report detailed information about their surrounding physical environment. Large numbers of such devices can be deployed in areas of interest (like inaccessible terrains, disaster-struck areas, or embedded in our everyday environment) and form a sensor network using self-organization and collaborative methods. Each one of these devices carries a number of sensors and a transceiver, enabling it to sense its environment and to communicate with the other network nodes, respectively.

Because of their nature, i.e., the inherent restrictions in energy, cost and processing power, wireless sensor networks nodes are more prone to hardware failures and adversarial effects than nodes participating in other wireless networks, and should also consume considerably less energy, in order to operate for very long periods of time. Furthermore, these networks are expected to be deployed in “difficult” or even hostile environments, so fault tolerance, low energy consumption and obstacle presence should be taken into consideration. All of the above characteristics and restrictions make the design of protocols for wireless sensor networks a real challenge.
The flexibility, fault tolerance, high sensing fidelity, low cost and rapid deployment features of wireless sensor networks helped to create many new and exciting application areas for remote sensing. This wide range of applications is based on the use of various types of sensors (i.e., thermal, acoustic, magnetic, etc.), in order to monitor a wide variety of conditions (e.g., temperature, object tracking, humidity) and report them to one or more control centers (fixed or mobile). This control center could be some human authorities responsible of taking action upon the realization of some crucial event or a server forwarding readings from the sensor network to the Internet for further use. Thus, sensor networks can be used for important applications, including (a) environmental applications (such as fire detection, flood detection, precision agriculture), (b) health applications (like telemonitoring of human physiological data) and (c) home applications (e.g., smart environments).

As time goes by, the initial vision of sensor networks is extended to include additional characteristics, such as heterogeneous sensor networks, i.e., networks consisting of nodes that have different resources (processing power, energy supplies, etc.), or networks with actor nodes, i.e., nodes that carry actuators apart from sensors. For a basic survey of wireless sensor networks, see [1].

1.1 The Problem – Motivation

As mentioned above, a wireless sensor network consists of a large number of nodes that are usually scattered in an area of interest and form a sensor network, as shown in Fig. 1. Each of these scattered nodes has the capability to locally collect and route data to one or more control centers. Data is propagated to the control center using a multi-hop data dissemination protocol running on every sensor network node.

Let’s assume the realization of a series of \( K \) crucial events \( E_i \), with each event being sensed by a single node \( p_i \) \((i = 1, 2, \ldots, K)\). Then, we define the multiple event propagation problem as follows: how can each node \( p_i \), via cooperation with the rest, propagate the information \( \text{info}(i) \) regarding the event \( E_i \) to the control center of the network in an efficient and fault-tolerant way.

Due to the nature of wireless sensor networks, i.e., the very large number of nodes, real network deployments are expensive and difficult to maintain, and this is one of the reasons few real large sensor network deployments exist up to date. Although such deployments are crucial for testing accurately the actual performance of the proposed network protocols, wireless sensor network simulations provide many advantages over real world deployments, like e.g., the ability to easily test a great variety of parameters in protocols and deployment settings. Simulation also provides researchers with a number of other significant benefits, including repeatable scenarios, isolation of parameters, and exploration of a variety of metrics. It is generally not feasible, or it is too strenuous to run such extensive tests in this manner over a real wireless network. Furthermore, simulation represents a very useful complement to rigorous analysis of algorithms, in the context of computational complexity; in order to get asymptotic results and investigate network scalability, such analysis is performed under simplified models that do not capture important aspects, such as detailed technical specifications and network conditions. Because of the importance of simulation in wireless sensor networks, there rises a need for the use of more realistic network models while simulating such networks than the ones currently used.
A very large part of the simulations included in research papers dealing with wireless sensor networks make a lot of simplifying assumptions that lower the overall simulation complexity, in order to conduct experiments with a lot of nodes in reasonable time. Even though there has been a vast amount of research in simulating wireless sensor networks, one such common simplification is that the majority of the network models used assume unobstructed areas, i.e., areas without any obstacles present in the network deployment area; mainly implicit modeling assumptions on obstacles (i.e., on the node density) have appeared and such assumptions are certainly unable to model real obstacles decently. It is our belief that the inclusion of obstacles has a great impact on both the design of protocols for wireless sensor networks, as well as on the simulation and evaluation of these protocols’ performance. In other words, in order to be a bit more realistic, since wireless sensor networks are expected to be deployed in hostile environments and/or outdoor areas with harsh environmental conditions, obstacles should be taken explicitly into consideration while designing a protocol for such networks and also while evaluating its performance via simulation. Furthermore, and in addition to physical obstacles, virtual obstacles, like the local absence of functioning nodes, can dynamically emerge during the network’s operation (e.g., when the energy of sensor nodes in an area runs out, etc.).

The need for an obstacle model specifically for sensor networks rises from the special characteristics these networks have. Because of the inherent locality and simplicity of sensor networks’ protocols, the constrained resources in the network and the lack of global network knowledge, the presence of obstacles is expected to affect protocol performance (and even correctness) in a number of ways and should be taken seriously into account. In this sense, a model adequate for simulating ad hoc networks may be unsuitable for wireless sensor networks, or may need additional features to produce good results.

Especially in the case of greedy geographic routing algorithms for sensor networks, the use of obstacles in simulations should, in many cases, have a large impact and could reveal inherent protocol design weaknesses, that might be difficult to discover otherwise. Routing protocols that use geolocation information to make forwarding decisions comprise a large part of the proposed routing algorithms for sensor networks. This is due to the fact that usually such information is assumed to be available in sensor network nodes, since much of the information extracted from these networks is location-specific. Another nice feature that these protocols possess is that they require only a small table containing information about their neighbors, instead of using big routing tables. As an example, think of a sensor network with one control center and a large wall inside the deployment area that cuts off communication from most sensor nodes to the control center. A pure greedy forwarding scheme, i.e., forwarding data only if a neighbor nearer to the control center exists, used inside such a deployment field will have little chance of overcoming this obstacle. Also, pure geographic routing algorithms could be trapped into dead ends formed by obstacles, as will be explained in the following section.

In this work, we want to emphasize the usefulness of including obstacles in sensor networks simulations, and not to propose a very realistic simulation model in order to produce hands down realistic results. Our obstacle model, apart from giving insight to effects of obstacle inclusion in routing protocol performance, is basically useful in detecting weaknesses in routing protocol design and discovering situations where routing protocols may fail.

2 Related Work and Our Contribution

2.1 Models for Obstacles inside Wireless Sensor Networks

Although obstacle avoidance is a well studied subject in distributed computing in general, there has not been much attention toward the modeling of obstacles in wireless sensor networks particularly, and also toward studying explicitly and systematically how the various data propagation protocols behave against obstacles in the network deployment area. A summary of some obstacle-related models is given in this section.

2.1.1 Obstacles modeled as polygonal objects

There is some related work concerning the placement of obstacles inside the network deployment area in mobile ad hoc networks. An obstacle mobility model for mobile ad hoc networks is proposed and evaluated in [2] and [3], versus the random way-point and other more commonly used mobility models for ad hoc networks. This model uses polygonal obstacles that may be present inside the network deployment area, influencing the way mobile nodes move, and also
wireless transmissions between nodes. The obstacles model buildings and other structures that provide barriers to the movement of mobile nodes, thus being more realistic than other models. Obstacles have also an effect on the communication between nodes, the extent of which depends on the wireless channel model used in the simulations (from simple line-of-sight blocking to more complex signal propagation models).

This approach is practical, but a little too general, since it does not provide specific obstacle templates to be used in simulations. Also, it is better suited for ad hoc networks and not wireless sensor networks, since it does not cater for the cases that there are regions within the deployment area that have very few or no sensors at all, forming a virtual obstacle. This case is quite common in sensor networks for a number of reasons: nodes use up their energy sources, are destroyed by external factors, etc. Finally, in this model there is no reference to obstacles that have limited lifetime or make their appearance after the network has begun functioning. So, all obstacles are considered static and deterministic in nature. In [2] the obstacle map used as an example was a projection of some of the buildings in the campus of University of California at Santa Barbara.

2.1.2 Covering randomly the network area with obstacle squares

Another approach to modeling obstacles in the environment is to divide the deployment area into equal-size square cells. Each cell is either occupied by an obstacle (obstacles are static and do not move) or not. The obstacle density is known in advance and obstacles are placed in a random fashion, by assigning an independent probability to each cell as to whether it will be occupied by an obstacle, or this probability is relative in some way, leading to clusters of cells occupied by obstacles (in order to produce bigger obstacles).

This approach is useful and easy to implement while simulating wireless sensor networks, but in general is not as realistic as the other models presented in this section. Moreover, the shape and size of the obstacles may also have an effect on the network, and in this specific model such features are neglected. Also, while this is a probabilistic model for creating obstacles, all obstacles are created before the network starts functioning.

2.1.3 Obstacles modeled as routing holes

An interesting approach is described in [4], where all obstacles are modeled as routing holes, i.e., regions without enough sensors to propagate the information closer to its destination. Generally, routing holes are defined as simple regions enclosed by a (possibly concave) polygonal cycle which contains all the nodes where information can be stuck (and subsequent forwarding is ceased).

This definition of obstacles as routing holes is very general and can have many interpretations. Apart from the case of communication voids (areas with no or very few sensor nodes), the meaning of routing holes can be defined by specific application needs, or indicate geographical characteristics of the sensor deployment area. E.g., in a forest fire detection application, the sensors inside the fire region will be destroyed and leave a routing hole in the network. Identifying the area that is covered by the routing hole equals to detecting the area already covered by the fire. Massive drop-off of sensors on irregular terrains leaves holes that correspond to mountains or shadows. Holes can also be used to detect regions with low sensor density due to the depletion of the nodes’ energy supplies. Therefore holes can be thought of as the areas where adding new nodes will significantly improve the overall connectivity of the network.

Although this approach gives an insight to the significance of obstacles in wireless sensor networks, and also what could be modeled as an obstacle, it is a bit general and gives no reference obstacles to use in simulations. This is critical, since in simulations there is always the need of repeating experiments and testing a number of methods against each other. It is, however, useful as a theoretical analysis tool.

2.1.4 Obstacles modeled as straight walls

Another approach to incorporating obstacles in wireless sensor network simulations can be found in [5]. Obstacles are modeled as straight walls that are parallel to the $x$ or the $y$ axis with varying length and number (this notion of obstacles modeled as walls is also used in [6]). Furthermore, apart from the walls case, the case of voids, i.e., areas of regular shape which contain no nodes, is explored in [5]. However, all obstacles are placed before the network starts to function, and so communication voids that appear during the operation of the network cannot be set explicitly.
A similar approach is followed in [7], where obstacles (walls) are of fixed length, the center of each wall is positioned randomly in the network field, and it is equally possible for each obstacle to be parallel to the $x$ or the $y$ axis. In [25] the authors considered one wall in the middle of the network. Its length was varied as 0.5, 1 and 2 times the nodes’ radio range.

This approach of modeling obstacles is useful, especially in the case of only one control center in the sensor network and geographic routing (because in this case walls effectively block greedy forwarding schemes), but is simplistic, and does not take in account parameters as the obstacle size, shape, etc. This is important, because routing protocols may perform differently in scenarios with different types of obstacles, as explained in the following model.

2.2 Solutions to the obstacle avoidance problem in Wireless Sensor Networks – Simulation studies

In order to simulate the effect of obstacles on the communication between nodes inside the deployment area, the authors of [2] implemented a number of extensions to the GloMoSim and ns-2 network simulators, which provide a mechanism for the placement of obstacles within the simulation terrain, with which the user can define the position, shape and size of these obstacles. They initially utilized a calculation of the signal-blocking regions in which the wireless signal is blocked due to the obstacles present in the deployment area and in [3] the authors used a signal propagation model that simulates the fading of a radio signal as it propagates through obstacles that lie between a pair of nodes. A series of experiments was conducted using again the GloMoSim and ns-2 network simulators, and their results in both works indicate that the inclusion of obstacles in their network model has a significant effect on the performance of the ad hoc protocols they chose to study. This is in part of course due to the fact that their mobility model considers obstacles as well, whereas other commonly used mobility models do not.

Generally, the proposed solutions for providing obstacle avoidance in wireless sensor networks have some phase for locating the obstacles inside the sensor network deployment area, and use this information to build a network topology improving network connectivity. A large part of such algorithms uses face routing on a planar subgraph, such as the algorithm in [8]. Depending on the approach, this can happen as a setup phase before the network starts operating, or during its operation.

In [4] the authors present a solution to identifying routing holes, using a greedy algorithm, called BoundHole. The main idea when encountering a situation where information cannot be propagated further toward its destination while using a greedy logic, is to find a directed closed cycle that encloses the routing hole. This cycle starts from the node where the forwarding stuck and following a local rule at each node goes back to the same node, identifying the obstacle. Assuming geolocation capabilities, this algorithm is local and simple, and each node stores information only about its one-hop neighbors.

After identifying the routing holes inside the network, and while using greedy forwarding the information under propagation gets stuck at a node, the routing scheme scheme changes and the boundary of the hole is chosen as the propagation path. When a node that is closer to the destination than the original stuck node is encountered, greedy forwarding is chosen as the forwarding scheme again.

Another solution, without assuming geolocation capabilities (no GPS), is virtual coordinates [5]. The authors introduce a protocol for assigning virtual coordinates to nodes, that reflect network connectivity, in order to use a greedy forwarding scheme. The algorithm for assigning these coordinates is iterative. Different geolocation capabilities assumptions lead to different protocol complexity results (more nodes with GPS capabilities, less iterations to reach the best solution to the problem and message broadcasts). A simplified version of this algorithm (as the original version was deemed too complex) was implemented in a real environment and tested while placing some small obstacles in the network field in [9].

Also, [10] proposes trading-off the cost for obtaining a limited extra knowledge of the network with the benefit in subsequently optimizing routing toward the control center using this extra knowledge. This idea, although general, can be used in order to locate obstacles early enough and adjust accordingly. Finally, in [11], the obstacle model which we present here, has been used to evaluate the performance of a geographic routing protocol under “difficult” network conditions.
2.3 Other related work

In addition to modeling the obstacles that are present in the network area, there is also a great deal of related work in terms of signal attenuation and fading due to the presence of obstacles [12]. In a wireless network deployment area with obstacles present, it is obviously possible for two network nodes to be able to communicate directly via multipath signal propagation with each other, even though an obstacle is blocking the line of sight between them. Mechanisms such as reflection, distraction and scattering have been studied both analytically and experimentally. A number of models for signal propagation have been proposed, to enable calculation of the strength of the received signal from a source to a point in the network field. These models use formulas which include constants that depend on a number of parameters, such as the frequency of the radio signal, the surrounding environment, the size of the antennas, etc., which are calculated experimentally. Also, in some network simulators, like TOSSIM [13], obstacles can be described indirectly by specifying available links between pairs of nodes, but no specific model is provided to the user. In [14] and [15] the design and implementation of the SwimNet network simulator has been presented. This simulator concerns wireless networks in general and proposes a scalable parallel simulation testbed for wireless and mobile networks, in contrast to our work which focuses on sensor networks, and especially on the presence and impact of obstacles.

In general, this work focuses on giving models for average use cases (e.g., indoor office spaces) and does not consider explicitly the effect of the various physical obstacles, or obstacles that are formed out of node distribution patterns. Although using such models may be more realistic on the average case, it is not easy to trace specific protocol weaknesses due to obstacles. Also, in the case of simulations, the adaptation of such schemes introduces additional computational overhead, which may lead to prohibitive execution time for experiments with a very large number of nodes.

2.4 Our Contribution

We propose a systematic and generic obstacle model to be used in simulations of wireless sensor networks and we provide a categorization of the obstacles’ types, based on a variety of criteria. We believe that the inclusion of obstacles in wireless sensor network simulators will lead to interesting and important findings and that the categorization of obstacles is necessary in order to study the effect of the various types of obstacles in the behavior of data dissemination protocols for wireless sensor networks. Our model caters for both physical and “communication” obstacles, deterministic and probabilistic ones (defined in Sec. 4). Furthermore, we include obstacles of various shapes that are expected to appear in more realistic deployment scenarios. Also, obstacles of various shapes in our model can be combined to produce more complex shapes.

Some reasons to use our obstacle model are the following:

- Such a model can be easily implemented and used.
- It covers a number of different obstacle types.
- Can be easily randomized and used to produce random network instances in a network simulator.
- Probabilistic obstacles may be used to produce more realistic simulations, and can also be useful in cases like delay tolerant networks.
- Produce more realistic network topologies, instead of assuming homogeneous randomly deployed nodes.

We have implemented our model of obstacles in the simDust network simulator [17–21, 24], in order to incorporate the proposed obstacle model to this simulator. In this way, we created a simulation environment that integrates a variety of network topologies, protocols and obstacles. We provide experimental results comparing the performance of several representative protocols for data propagation in wireless sensor networks in various settings of obstacles and study their behavior. Our findings demonstrate the crucial impact of obstacles in protocol performance in general, as well as the particular effect of certain obstacles to each protocol.

Our main goal in this paper was to propose and implement a new model for obstacles and also investigate its suitability and applicability to the evaluation of the performance of protocols used in wireless sensor networks, under
the presence of obstacles. We focused on the problem of data propagation (routing), since it is severely affected by obstacles and also because routing functionality is fundamental for higher level services and applications. Because of this fact, we believe that our findings provide significant insight to the impact of obstacles on other problems as well; we plan to use our model to explicitly investigate this impact.

Regarding routing, we chose to evaluate a few protocols that evolve in different, characteristic ways (i.e., a greedy, single-path protocol and its adaptive extension, as well as an optimized redundant protocol that uses multiple paths). Our purpose was to investigate whether our framework is able to demonstrate the different impact of various obstacles to each protocol. Other routing protocols from the state-of-the-art could have served this purpose equally well. Again, we plan to investigate the behavior of other protocols in our future work.

This paper is an extended version of the work included in [26]. We have extended the motivation section in the introduction, describing the necessity of using obstacles particularly in sensor network simulations, especially in the case of greedy geographic routing algorithms. The related work section has been rewritten, and now includes a detailed presentation of all obstacle models (to our knowledge) previously used in simulations of wireless sensor networks. Each model is presented and analyzed, along with its advantages and its shortcomings. Finally, the section describing simulation results features additional figures, which describe experimental results concerning multiple circular obstacles and also some extra discussion on the interpretation of these results.

3 A Model for Sensor Networks

Based on the technological specifications of existing wireless sensor systems, each node is a fully-autonomous computing and communication device with constrained resources, equipped with a set of monitors (e.g., sensors for temperature, humidity, etc.) and characterized mainly by its available power supply (battery) and the energy cost of computation and transmission of data. The communication equipment broadcasts messages to nearby devices within transmission range \( R \), and can also use a directed transmission of angle \( \alpha \) around a certain line (possibly using some special kind of antenna), see Fig. 5. The transmission range \( R \) can vary (i.e., the transmission power can be set at appropriate levels), while the transmission angle is fixed and cannot change throughout the operation of the network (since this would require a modification or movement of the antenna used). Note that the protocols considered in this work (see Sec. 5) can operate even under the broadcast communication mode (i.e., \( \alpha = 2\pi \)).

We consider a simple sensor network for the remote surveillance of a region or for data collection in an ambient intelligence setting. In practice, such a network may consist of several hundreds or thousands of sensor devices deployed within that region. Let \( n \) be the total number of sensor devices, that are present in an area of size \( A \). In some cases, the devices may be deployed in a regular fashion (e.g., a 2-dimensional lattice, or a linear array) within that region. More generally, however, communication and networking protocols cannot assume structured sensor fields. In our remote surveillance network we assume that the sensor devices do not move and that they are not able to change their physical position.

A particular user of this remote surveillance system, which we call the control center \( S \) and which is not resource-constrained like the other nodes of the network, may contact the sensor devices (e.g., via a long-range radio link) in order to acquire information regarding the environmental conditions. In this sense, the user injects sensing tasks in the network, i.e., by broadcasting messages with a task description; the system can support a variety of task types [16]. Those sensor devices that match the task description report to the control center using multi-hop wireless communication and routing mechanisms described in Sec. 5. In this work, we assume a single, static (not mobile) control center.

In this paper we focus on some of the physical characteristics of the deployment area and their effect to the performance of the data propagation mechanism used. Our approach addresses some fundamental issues that are present in sensor networks and is presented in detail in the following section.
4 A Model for Obstacles

4.1 Physical and Communication Obstacles

Initially, we define two classes of obstacles: the class of physical obstacles and the class of communication obstacles.

Definition 1 (Physical Obstacle $O^{\text{phy}}$) A physical obstacle corresponds to an obstacle that prevents the physical presence of sensor devices – a device that is positioned over the obstacle is automatically “destroyed”. In this sense, the network area that is occupied by the obstacle is virtually empty of sensor devices.

Definition 2 (Communication Obstacle $O^{\text{com}}$) A communication obstacle corresponds to an obstacle that causes disruption to the wireless communication medium. In particular, we assume that if the line of sight between two devices crosses the obstacle, then communication is blocked. So, even though some sensor devices may be deployed on top, or even inside the obstacle, no communication with other devices can take place.

Essentially, the class of physical obstacles relates to situations where there is a lack of sensor devices in specific parts of the area of deployment, i.e., the local network density is zero. Those devices that are on the boundaries of the physical obstacle are still able to communicate as long as they can overcome it by possibly increasing the transmission power. On the other hand, for the class of communication obstacles, although some sensor devices can be located inside (or on top of it) the area affected, no communication with the rest of the network is possible. Of course, an obstacle can fall within both classes, blocking any physical presence of devices and communication activity.

4.2 Obstacles’ shape

We present now a collection of geometric elements that can be used to describe an obstacle’s shape. These basic elements can be used to represent simple real world objects, or can be combined to produce more complex objects (see Fig. 4).

**Rectangular (Orthogonal) Obstacles:** this type of obstacles roughly corresponds to buildings and large vehicles. Obstacles of this shape are not supposed to be too big, compared with the overall network plane dimensions. Obstacles of this type can be found mostly in urban environments.

Definition 3 (Rectangular Obstacle $O_{\text{rect}}(p, l, w)$) A rectangular shaped obstacle is positioned on point $p$ of the deployment area with dimensions $l \times w$ ($l$ stands for length and $w$ for width). The obstacle’s position is defined in terms of its upper left corner.

**Circular Obstacles:** this type of obstacles roughly corresponds to craters, large rocks, lakes, ponds and (big) tree logs. Objects of this type can be found in outdoor environments, countryside, battlefields, etc.

Definition 4 (Circular Obstacle $O_{\text{circ}}(p, r)$) A circular shaped obstacle is centered on point $p$ of the deployment area with radius $r$.

**Crescent (Boomerang) Obstacles:** this type of obstacles roughly corresponds to a lake with a shape that resembles that of a crescent or a boomerang. Such an obstacle is defined by a circle and an ellipse, as shown in figure 2. Although obstacles of this exact shape may be hard to encounter in real environments, they present a challenge for routing protocols. This is due to the fact that such an obstacle formulates a “loose” dead-end, in the sense that data propagation tend to be trapped in the concave part of the crescent.

Definition 5 (Crescent Obstacle $O_{\text{cres}}(p, r, s)$) A crescent shaped obstacle is centered on point $p$ of the deployment area with radius $r$ of the circle in which the obstacle is inscribed and width $s$ of the enclosing ellipse.

**Ring Obstacles:** this type of obstacles can represent areas of the network that are somewhat isolated from the rest of the nodes, i.e., the nodes on the inside of the ring are separated from the rest of the network by the outer part of the ring obstacle.

Definition 6 (Ring Obstacle $O_{\text{ring}}(p, r, m)$) A ring shaped obstacle is centered on point $p$ of the deployment area with radius $r$ of the outer circle and radius $m$ of the inner circle.
Figure 2: A crescent-like obstacle  
Figure 3: A ring obstacle  
Figure 4: Combination of obstacle shapes in order to represent complex obstacles

**Stripe Obstacles:** this type of obstacles roughly corresponds e.g., to a river (communication obstacle) crossing the network deployment area or a long wall (physical obstacle) situated in the network deployment area. The main difference from the rectangle obstacles is their size, i.e., they are much bigger than rectangle obstacles should be, and the way these obstacles are defined by the user.

**Definition 7 (Stripe Obstacle $O_{\text{ring}}(p, w, a)$)** A *stripe shaped obstacle* is positioned on point $p$ of the deployment area with width $w$ and angle $a$ with respect to the horizontal boundary of the area. The obstacle’s position is defined in terms of its upper left corner.

### 4.3 Stochastic Presence of Obstacles

The third criterion in our obstacle model is whether *randomness* characterizes the presence of obstacles:

**Deterministic Obstacles:** this category consists of all the obstacles that are present throughout the duration of an experiment (i.e., from the beginning to the end) and do not change in any manner. They are defined from the user before the beginning of the experiment. Deterministic obstacles can belong to any type and shape of obstacles, as described previously.

**Probabilistic Obstacles:** this category, on the contrary, consists of obstacles that are not present throughout the duration of a simulation experiment, but appear in a random fashion for a period of time and even disappear. For example, think of a train passing through the network deployment area, or a even a road inside the deployment area that is crossed by cars. These obstacles have a temporary effect on the nodes that are situated in the area they appear. Those nodes cease to function throughout the life span of the probabilistic obstacle by which they are capped. Furthermore,
stochastic obstacles may capture areas whose density drops significantly over time (due to physical faults, permanent or temporary, as well as software decisions, i.e., power saving schemes that put sensors to sleep). Probabilistic obstacles can also belong to any type and shape.

In order to define a probabilistic obstacle, a simple definition procedure is as follows: the user first defines the type and shape of the obstacle, and then assigns to it a start time and a duration. The duration can be defined to be till the end of the simulation.

5 Protocols for Data Propagation

In this paper, we focus on distributed algorithms for the network layer. We present below three representative (in terms of their logic) protocols from our previous research work, that try to avoid flooding the network, achieving good performance (with respect to time and energy) and robustness. All three protocols use to some extent geographic information to make (greedy) forwarding decisions.

5.1 The PFR Protocol

The Probabilistic Forwarding protocol (PFR) [19] is inspired by the probabilistic multi-path design choice for Directed Diffusion[23]. The basic idea of PFR is to minimize energy consumption by probabilistically favoring certain paths of local data transmissions toward the control center.

The protocol avoids flooding by favoring (in a probabilistic manner) data propagation along sensor nodes which lie “close” to the (optimal) transmission line, \( ES \), that connects the node detecting the event, \( E \), and the control center, \( S \). This is implemented by locally calculating the angle \( \phi = (EPS) \), whose corner point \( P \) is the sensor node currently running the local protocol, having received a transmission from a nearby node, previously possessing the event information. If \( \phi \) is equal or greater to a predetermined threshold (\( \phi_{\text{threshold}} \)), then \( P \) will transmit (and thus propagate the event information further). Else, it decides whether to transmit with probability equal to \( \frac{\phi}{\pi} \). Because of the probabilistic nature of data propagation decisions and in order to prevent the data propagation process from early failing, we initially use (for a short time period which we evaluate) a flooding mechanism that leads to a sufficiently large “front” of sensors possessing the data under propagation. When such a “front” is created, we perform probabilistic forwarding.

Note that transmission along this line is energy optimal. However, it is not always possible to achieve this optimality, for a variety of reasons. Essentially, PFR captures the intuitive, deterministic idea “if my distance from \( ES \) is small, then send, else do not send”. This idea was enhanced by random decisions (above a threshold) to allow some local flooding to happen with small probability and cope with local sensor failures.
5.2 The LTP Protocol

In this subsection we give a short description of the Local Target protocol (LTP) [21]. The basic idea is to try to search for all active neighboring nodes and then use the information retrieved in order to forward (i.e., propagate) the data toward the neighbor that is closer to the control center. In this protocol, each node $p'$ that has received $info(E)$ from $p$ (via, possibly, other nodes) does the following:

**Phase 1: The Search Phase.** It uses a periodic low energy broadcast of a beacon in order to discover a node nearer to control center than itself. Among the nodes returned, $p'$ selects a unique node $p''$ that is “best” with respect to progress toward the control center, that is, the node $p''_E$ that among all nodes found achieves the bigger progress on the $p''S$ line, should be selected.

**Phase 2: The Direct Transmission Phase.** Then, $p'$ sends $info(E)$ to $p''$ and sends a *success* message to $p$ (i.e., to the node that it originally received the information from).

**Phase 3: The Backtrack Phase.** If consecutive repetitions of the search phase fail to discover a node nearer to control center, then $p'$ sends *fail* message to the node that it originally received the information from.

In the above procedure, propagation of $info(E)$ is done in two steps; (i) node $p'$ locates the next node ($p''$) and transmits the information and (ii) node $p'$ waits until the next node ($p''$) succeeds in propagating the message further toward the control center. This is done to speed up the backtrack phase in case $p''$ does not succeed in discovering a node nearer to control center.

5.3 The VTRP Protocol

The Varying Transmission Range (VTRP) [17] basically works in a *search and forward* way similar to LTP, and also by varying the range of transmissions in order to achieve better performance, compared to fixed transmission range data propagation, in some rather frequently occurring situations like: (a) the case of low densities of sensor nodes, and (b) because of the possibility to increase transmission range, VTRP performs better in cases of obstacles or faulty/sleeping sensors. Also, it bypasses certain critical sensors (like those close to the control center) that tend to be overused, thus prolonging the network lifetime.

When a node $p'$ receives some $info(E)$ from node $p$, VTRP works in three phases, of which the first two are identical to LTP's first two phases.

**Phase 3: The Transmission Range Variation Phase.** If phase 1 fails to discover a node nearer to control center, $p'$ enters the *transmission range variation phase*. More specifically, each node maintains a local counter $\tau$, with initial value $\tau = 0$. Every time the search phase fails, this counter is increased by 1. Thus $\tau$ is an indication of the number of failures to locate an active node. Based on $\tau$, the node modifies its transmission range $R$ according to a change-function $F(\tau)$.

We consider here only one of the originally four different change-functions for varying the transmission range defined in [17] – that of *Multiplicative Progress*. In this case, the transmission range of the node is increased more drastically.

$$F(\tau) = R_{new} = R_{init} + R_{init} \cdot m \cdot \tau,$$

where $m$ is a small constant, ($m = 3$).

This relatively drastic change leads to bigger probability of finding an active node, however it leads to higher energy consumption.

6 The Simulation Environment

This section provides a description of the components needed for conducting our simulation experiments, including the simulation environment, the metrics used to evaluate the experimental results and the simulation scenarios.

To evaluate the performance of the proposed protocols we conducted an extensive simulation analysis. In this work, we use simDust (see [17–21, 24]), a lightweight network simulator. simDust is implemented using C++ in Linux, with the aid of the LEDA library, that provides efficient data types and visualization capabilities. We chose C++ for its
efficiency and LEDA because it provides a large number of efficient data structures using a unified interface, making their use simple enough for our purposes.

In contrast to other more detailed network simulators like the Network Simulator, simDust makes an abstraction of the physical and MAC layers. We make the assumption that all packet collisions and other related issues are dealt with in an underlying MAC layer, and do not take them into account in our simulations. Although this is fairly simplifying and simDust cannot provide detailed measurements on parameters such as the number of dropped packets or the precise execution time, this abstraction of the lower network levels allows simDust to reduce the execution time of simulation experiments, enabling the study of very large network instances (in the scale of tens of thousands).

One basic idea in this simulator is that a number of general classes are provided to the programmer to implement new routing protocols for wireless sensor networks. The whole network operates in a synchronized fashion, specifically in what we refer to as simulation rounds. In each round, each node of the network can do some internal processing, send or receive messages, or sleep (i.e., turn off transceivers and CPU). The programmer defines what each node is supposed to do in the course of these simulation rounds, i.e., what to do if an event occurs, if a message is received, etc. Events in the network can be triggered by the simulator, that cause nodes in the area of the event to produce messages that need to be forwarded to the control center of the network. In the current implementation of simDust, each event is detected by only one node. Nodes are placed in a rectangular shaped field, with dimensions defined by the programmer. Nodes are placed using some random uniform distribution, or placed in a grid.

Regarding connectivity issues inside the network, each node maintains a list of its neighbors, i.e., the nodes that are inside a disk with the current node in the center and with a radius equal to its transmission range. We make the assumption that if a node is inside this disk will receive all messages transmitted from the current node. The transmission range of each node can change independently, and in this case neighborhoods are recalculated in the next simulation round.

Apart from the simulation part of simDust, there is also a visualization part that enables the programmer to have a visual understanding of the simulated network’s operation. The programmer, using the class defined for the data propagation algorithms and extending them to use some data types provided by LEDA, can produce a visualization of the network’s inner workings. This capability was implemented to provide easier understanding of each algorithm’s functions, and check the validity of our implementations.

Regarding the energy consumption of the devices, simDust implements a rather detailed energy cost model that enables relatively detailed measurements of the energy dissipation. Generally, each node in the sensor network can be in one of three different modes at any given time, regarding its energy consumption. These modes are: (a) transmission of a message, (b) reception of a message, and (c) sensing of events.

Following [22], for the case of transmitting and receiving a message we assume the following simple model where the radio dissipates $E_{elec}$ to run the transmitter and receiver circuitry, and $e_{amp}$ for the transmitter amplifier to achieve acceptable signal to noise ratio. We also assume an $r^2$ energy consumption when transmitting a signal with a range of distance $r$. Thus, to transmit a $k$-bit message at distance $r$ in our model, the radio expends

$$E_T(k, r) = E_{T-elec}(k) + E_{T-amp}(k, r)$$

and to receive this message, the radio expends

$$E_R(k) = E_{R-elec}(k)$$

$$E_R(k, r) = E_{elec} \cdot k$$

where $E_{T-elec}$, $E_{R-elec}$ stand for the energy consumed by the transmitter’s and receiver’s electronics, respectively. Concluding, there are three different kinds of energy dissipation which are:

- $E_T$: Energy dissipation for transmission.
- $E_R$: Energy dissipation for receiving.
- $E_{idle}$: Energy dissipation for idle state.
For the idle state, we assume that the energy consumed for the circuitry is constant for each time unit and equals $E_{elec}$ (the time unit is 1 simulation round).

As for the blocking of signal transmission between nodes, recall that we assume that two nodes can communicate directly (that is, if one node is in the communication range of the other) only in the case that there exists a line-of-sight path between them. If no line-of-sight exists between two random nodes, then they cannot communicate directly. In our model, line-of-sight is blocked only by physical obstacles.

Regarding the implementation of obstacles in our simulator, it was partially based on some data types provided by LEDA, and particularly the Circle and Segment data types, that correspond to circle shapes and line segments, respectively, and simplify the procedure of allocating space to each obstacle and positions for the nodes inside the network area. They also simplify the procedure of detecting whether an obstacle blocks the line-of-sight between two random nodes in the network. Obstacles are defined in text files that serve as input to the simulator. Such a text file defines the type of the obstacle, its dimensions, etc. Each such file can contain multiple obstacles.

In order to check the availability of line-of-sight between two random nodes, we can define a line segment for each two such nodes, and perform a check on whether this segment crosses any obstacle. If so, we decide that there is no line-of-sight between the two nodes and that they are not able to communicate with each other directly. This check is performed for each pair of nodes and for each obstacle in the network field, in order to determine the network neighborhood of each node (i.e., the set of nodes that can be contacted directly by the specific node.

Moreover, we make the assumption that the control center is located on the middle of the right edge of the network field. Because of this assumption, the direction of crescent obstacles in the network field is always the same.

Regarding the simulation parameters used for each experiment, the programmer defines the total number of nodes, the data propagation algorithm that will be used, the total simulation time (in rounds), the total number of events taking place inside the network, the possibility of an event happening in a random simulation round, and the parameters regarding the network field and node placement.

As for the output of the simulator, all results are gathered in text files that contain information both for the whole experiment and in greater detail (specifically, for each 10 rounds). The statistics that are noted down include, among others, success rates (events that have successfully reached the control center), the energy available in the network, the number of alive nodes, the total number of transmission and receptions, average number of hops to reach the control center, etc. Specifically, we explain a bit more about the metrics we used in simDust to evaluate the performance of the simulated protocols. As defined earlier, in the multiple event propagation problem we have a series of $K$ crucial events $E_i$ ($E_1, E_2, \ldots, E_K$) inside the network deployment area. Let $l$ be the number of events that were successfully reported to the control center $S$.

**Success Rate:** The success rate $P_s$ is defined as the fraction of the number of events successfully propagated to the control center over the total number of events, i.e., $P_s = \frac{l}{K}$.

**Total available energy:** A straightforward way to compare the performance of different protocols for wireless sensor networks is to study the available energy over time, both in each node and overall in the network. Let $E_i$ be the available energy for node $i$. We define the total energy available in the sensor network as $E_{tot} = \sum_i^n E_i$, where $n$ is the number of nodes in the network. Clearly, the less energy a protocol consumes, the better.

**Number and distribution of “alive” nodes:** Another metric of the protocols’ efficiency is the total number of “alive” nodes in the network and the way they are distributed. By “alive” nodes, at a certain point of time, we refer to the nodes that haven’t run out of energy supplies. As in the previous cases, the more alive nodes, the better. Furthermore, the distribution of these nodes is as important as their number, and particularly the condition of critical sensor nodes, such as nodes lying close to the control center.

### 7 Experimental Scenarios and Discussion of Results

This subsection provides a description of the setup used in our simulation experiments, regarding node density and distribution, event and obstacle generation. We give a short description of each set of experiments and then discuss the results of that specific set of experiments. All experiments were conducted in a network field with dimensions 1000 by 1000 units, using deterministic obstacles (we plan on conducting experiments using probabilistic obstacles in the future). We make the assumption that the lower left edge of the field has coordinates $(0, 0)$ and that the control center
Figure 6: Success rate of LTP, PFR and VTRP for various numbers of nodes \( n \in [500, 3500] \), with initial transmission range \( R = 50m \).

Experiments without obstacles: Our first set of experiments regards sensor networks of varying size (number of nodes), in a field with no obstacles. More specifically, we position \([500, 3500]\) nodes in the network field, in steps of 500 nodes, and generate 250 events each time to be propagated to the control center. Our main purpose for this set of experiments was to determine a network size at which all three protocols perform very well, in order to make a fair comparison for the following sets of experiments for all protocols. Our only criterion for this decision was the success rate of each protocol.

As it can be seen in Fig. 6, the performance of the three protocols is increasing as the network density increases, and also VTRP achieves a high success rate earlier than the other two protocols. Moreover, we notice that after the number of nodes of the network reaches 2500, all three protocols achieve very high success rate. For this reason, it is fair to make a comparison between these protocols under this condition \( n = 2500 \).

Experiments with Circular obstacles: For this set of experiments we created sensor networks with varying size, containing one circular obstacle in the center of the field, i.e., with coordinates \((499, 499)\). The area covered from this circular obstacle ranged from 5% to 70% of the total network area, in steps of 5%. The size of the sensor network for each obstacle size was varying, in order to maintain the same network density as in the case of no obstacles in the network. Also, our experiments were conducted for both cases of physical and communication circular obstacles.

For this set of experiments only one event was generated in each run of the simulator, more specifically in a node with coordinates \((0, 499)\), in order to investigate the case of the propagation of an event generated at a node placed diametrically to the control center. Also, each run of the simulator lasted for a specific number of simulator rounds, meaning a specific amount of time for which we wait for the report of the single event generated. This event setup was used for 100 runs of the simulator, with the position of the rest of the nodes in the field being recalculated each time.

The results for this set of experiments can be seen in Fig. 7. It is clear that LTP does not perform as well as the other two protocols, especially VTRP, which performs excellent in any obstacle size. PFR achieves good performance for many obstacle sizes, but after a certain point fails to propagate data to the control center. This is probably due to the fact that the obstacle blocks the creation of the protocol’s propagation front (as described in 5, and it cannot overcome the obstacle anymore. Moreover, communication obstacles prove to be harder to overcome than physical obstacles, as expected. Also, we notice that although we keep the network density unchanged, i.e., the connectivity of the network.
remains roughly the same, obstacles cause a significant effect to the protocols’ performance.

Figure 7: Success rate of LTP, PFR and VTRP in the presence of Circular Obstacles ($O_{\text{circ}}^{\text{phy}}(\{500, 500\}, \lambda)$) where $\lambda \in [126, 488]$ is the radius of the obstacle, for variable obstacle size, fixed node density and initial transmission range $R = 50m$.

Figure 8: Success rate of LTP, PFR and VTRP in the presence of Stripe Obstacles ($O_{\text{strp}}^{\text{phy}}(\{1000, 500\}, \lambda, -\pi)$) where $\lambda \in [50, 450]$ is the width of the obstacle, for variable obstacle size, fixed node density and initial transmission range $R = 50m$.

Experiments with Rectangular obstacles: As in the previous set of experiments, we created sensor networks with varying size, containing this time one rectangular obstacle in the center of the field, i.e., with coordinates $(499, 499)$. We considered two cases of rectangular obstacles, regarding their dimensions. In the first case, the length of the obstacle is the $\frac{3}{4}$ of its width and the area covered from it ranged from 5% to 50% of the total network area, in steps of 5%. In the second case, we created a thin rectangular obstacle, with a fixed width of 50 units and variable length, proportionate to the length of the network field, ranging from 5% of the field’s length to 70%, in 5% steps. In both cases, we used the same event setup as in the circular obstacle case. We tested both cases of physical and communication rectangular obstacles.

The results for this experiment set can be seen in Fig. 9. On the one hand, LTP starts dropping its success rate very early, due to the fact that when it reaches the obstacle it cannot find any nodes to forward data, both in physical and communication obstacles. On the other hand, PFR and VTRP, especially in the case of physical obstacles, achieve much higher success rates than LTP. Once again, the presence of obstacles in the network area has a significant impact to the operation of the sensor network, although the network density remains the same.

Experiments with Ring obstacles: For this set of experiments, we created a ring obstacle with its center in the center of the field, as in the previous cases. We studied three cases, regarding the width of the ring sector of the obstacle, setting it to 25%, 33% and 50% of the inner circle radius (see Fig. 3). In all three cases, we used the same event setup as in the circular obstacle case. We tested only the case of physical ring obstacles, as the communication obstacle case is essentially the same with the circular communication obstacle case.

The results for this set of experiments can be seen in Fig. 10. The general picture is the same for the three ring width cases, LTP relatively quick stops propagating data, PFR is fairly better than LTP, and VTRP achieves high success rate for all cases. From the point where the ring width goes beyond the fixed transmission range of LTP and PFR, this type of obstacle is identical to the physical circular obstacle case for these two protocols.

Experiments with Stripe obstacles: As mentioned earlier in the definition of stripe obstacles, an obstacle of this type corresponds to a river crossing the network field. We placed such an obstacle with its center in the center of the field, its length equal to the length of the network area and its width proportionate to the width of the network area.
Figure 9: Success rate of LTP, PFR and VTRP in the presence of Rectangular Obstacles ($O^{com}_{rect}(\{500, 500\}, \lambda, \frac{2}{3}\lambda)$ in the left, $O^{phy}_{rect}(\{500, 500\}, \lambda, 50)$ in the right), where $\lambda \in [274, 822]$ is the length of the obstacle, for variable obstacle size, fixed node density and initial transmission range $\mathcal{R} = 50m$.

Figure 10: Success rate of LTP, PFR and VTRP in the presence of Ring Obstacles ($O^{com}_{ring}(\{500, 500\}, \lambda, \frac{1}{4}\lambda)$ in the left, $O^{phy}_{ring}(\{500, 500\}, \lambda, \frac{1}{3}\lambda)$ in the middle, $O^{phy}_{ring}(\{500, 500\}, \lambda, \frac{1}{2}\lambda)$ in the right) where $\lambda \in [126, 488]$ is the radius of the circle enclosing the obstacle, for variable obstacle size, fixed node density and initial transmission range $\mathcal{R} = 50m$.

specifically, its width ranged from 5% to 45% of the network area’s width. As in previous sets of experiments, we used the same event setup as in the circular obstacle case. This set of experiments concerns only physical stripe obstacles.

The results for this set of experiments can be seen in Fig. 8. As expected, LTP and PFR do not perform well against stripe obstacles, because their fixed transmission range cannot overcome the gap created by the obstacle. Another interesting result is that VTRP does not achieve high success rate when the stripe obstacle reaches a certain size. This is due to the fact that the protocol requires a certain amount of time to alter the transmission range, which, apparently, is longer than the running time assigned to our experiment set.

Experiments with Crescent obstacles: Similar to the ring obstacle set of experiments, we placed a crescent obstacle in the center of the field and examined two cases, with the crescent width set to 75% and 50% of the obstacle’s radius (see Fig. 2). In all three cases we used the same event setup as in the circular obstacle case. In this set of experiments we tested only the case of physical crescent obstacles.

The results for this experiment set can be seen in Fig. 12. From this set’s results, it is apparent that this is the
Figure 11: Success rate of LTP, PFR and VTRP in the presence of multiple Circular Obstacles (two $O_{circ}^{phy}((500, 500), \lambda_1)$ in the left, three $O_{circ}^{phy}((500, 500), \lambda_2)$ in the right) where $\lambda_1 \in [89, 309]$ and $\lambda_2 \in [73, 230]$ is the radius of each of the circles in the network field, for variable obstacle size, fixed node density and initial transmission range $R = 50m$.

harder type of obstacle for all protocols to overcome, as it forms a “loose” dead-end. LTP, as expected, performs poorly and PFR’s performance drops very early, compared to the other obstacle cases. VTRP performs very well for the case where the crescents width is large, because it can overcome the obstacle by increasing the transmission range, but starts failing in the other case.

Figure 12: Success rate of LTP, PFR and VTRP in the presence of Crescent Obstacles ($O_{cres}^{phy}((1000, 500), \lambda, \frac{3}{2}\lambda)$ in the left, $O_{cres}^{phy}((500, 500), \lambda, \frac{1}{2}\lambda)$ in the right) where $\lambda \in [126, 488]$ is the radius of the circle enclosing the obstacle, for variable obstacle size, fixed node density and initial transmission range $R = 50m$.

Experiments with multiple Circular obstacles: We also conducted a number of experiments with 2 and 3 circular obstacles in the network field. The obstacles where placed with their centers in the line that is drawn from the control center to the node where the single event in the network is generated, and more specifically in (333, 500) and (667, 500) in the 2-circle case and (250, 500), (500, 500) and (750, 500) in the 3-circle case. We used the same event setup with the other experiment sets.
The results for this experiment set can be seen in Fig. 11. LTP fails to produce any positive success rate, while PFR’s performance is similar to that of the single circular obstacle. Also, VTRP performs worse than the single circular obstacle case. As we can see from this experiment set results, keeping the same network density and obstacle type (circular), while changing the number of obstacles in the field, produces totally different results.

This remark can be also clearly seen in Fig. 13, where we can see the performance of LTP and PFR for the various types of obstacles. The type and number of obstacles clearly affects the performance of protocols and also, network density is not directly related to the protocols’ success rate when there are obstacles present in the network field.

8 Concluding Remarks

In this work we have proposed a systematic obstacle model to be used in simulations of wireless sensor networks. We have also extended the simDust network simulator to incorporate this obstacle model and conducted a series of experiments to study the effect of obstacles in the performance of three representative protocols for wireless sensor networks. Our results indicate that the presence of obstacles in the deployment area of a wireless sensor network has, in certain cases, a significant impact on the performance of protocols for such networks. Furthermore, different obstacle shapes and sizes may affect each protocol in a different way; this provides useful information on which protocol is best to use for each obstacle case. Moreover, our results show that the effect of obstacles is not directly related to the density of such a network, and thus cannot be emulated by simply changing the density of the network.

Regarding our future work, we plan on implementing our model on other sensor network simulators, and also conduct simulation experiments with other routing protocols for sensor networks and study the effect of obstacles over their performance. In particular, it seems quite challenging to design protocols that handle efficiently the crescent and large rectangular type of obstacles. We also plan to extend our model and add an attenuation factor to obstacles. Mobility schemes are also considered as a means to extend our model. Finally, we are looking into using a plethora of experimental scenarios for our obstacle model and studying the effect of probabilistic obstacles in depth.

Acknowledgements

This work has been partially supported by the IST Programme of the European Union under contract number IST-2005-15964 (AEOLUS). Also, by the Programme PENED under contract number 03ED568, co-funded 75% by European Union – European Social Fund (ESF), 25% by Greek Government – Ministry of Development – General Secretariat
of Research and Technology (GSRT), and by Private Sector, under Measure 8.3 of O.P. Competitiveness – 3rd Community Support Framework (CSF).

**Short Author CVs**

**Ioannis Chatzigiannakis** is adjunct faculty member as assistant professor at the University of Patras, Computer Engineering and Informatics Department, Patras, Greece, and is also the managing director of Research Unit 1 (“Foundations of Computer Science, Relevant Technologies and Applications”) at the Research Academic Computer Technology Institute (RACTI), Greece.

**Georgios Mylonas** is currently a PhD student at the Computer Engineering and Informatics Department at the University of Patras, Greece, and is also a researcher at the Research Academic Computer Technology Institute (RACTI), Greece.

**Sotiris Nikoletseas** (corresponding author) is an Assistant Professor at the Computer Engineering and Informatics Department of Patras University, Greece, and was also a Visiting Professor of the University of Geneva, Switzerland. Adress: Sotiris Nikoletseas, Computer Technology Institute, N. Kazantzaki Str. 1, Patras University, 26500 Rion, Patras, Greece, email: nikole@cti.gr

**References**


