Implementation of 3D Obstacle Compliant Mobility Models for UAV Networks in ns-3

Paulo Alexandre Regis, Suman Bhunia and Shamik Sengupta
Department of Computer Science and Engineering
University of Nevada, Reno
Reno, NV, USA 89557
{pregis, sbhunia}@nevada.unr.edu, ssengupta@unr.edu

ABSTRACT

UAV networks are envisioned to play a crucial role in the future generations of wireless networks. The mechanical degrees of freedom in the movement of UAVs provides various advantages for tactical and civilian applications. Due to the high cost of failures in system-based tests, initial analysis and refinement of designs and algorithms for UAV applications are performed through rigorous simulations. Current trend of UAV specific simulators is mainly biased towards the mechanical properties of flying. For network-centric simulations, the intended measurements on the performance of protocols in mobile scenarios are conventionally captured from general-purpose network simulators, which are not natively equipped with comprehensive models for 3D movements of UAVs. To facilitate such simulations for UAV systems, this paper presents different mobility models for emulation of the movement of a UAV. Detailed description of three mobility models (random walk, random direction, and Gauss-Markov) are presented, and their associated movement patterns are characterized. This characterization is further extended by considering the effect of large obstacles on movement patterns of nodes following the three models. The mobility models are prepared as open-source add-ons for ns-3 network simulator.

CCS Concepts

•**Networks** → **Network simulations;**

Keywords

UAV, 3D, mobility model, obstacle, ad hoc network, ns-3

1. INTRODUCTION

With the advancement in the unmanned aerial vehicles (UAV) technologies, three-dimensional mobile networks are gaining popularity as 3D mesh networks. In addition to movement in a horizontal plane as in conventional mobile ad hoc networks (MANET), UAVs enable nodes to move in the vertical direction in 3D as well. The autonomous movement of UAVs in all directions makes it suitable for a wide range of operations, such as border surveillance, disaster monitoring, firefighter networks, relay for ground vehicles, tactical networks, imagery and sensing in rough terrain areas, and so forth [7]. Infrastructure networks are conventionally used to establish links with UAVs. But as the spatial diversity increases, the deployed infrastructure is not adequate to always associate UAVs with fixed access points. As a result, multi-hop mesh networks are gaining popularity. Data can be relayed over multiple UAVs to reach the destination without direct communication, either due to the link being blocked by a large obstacle, or the end nodes being out of communication range.

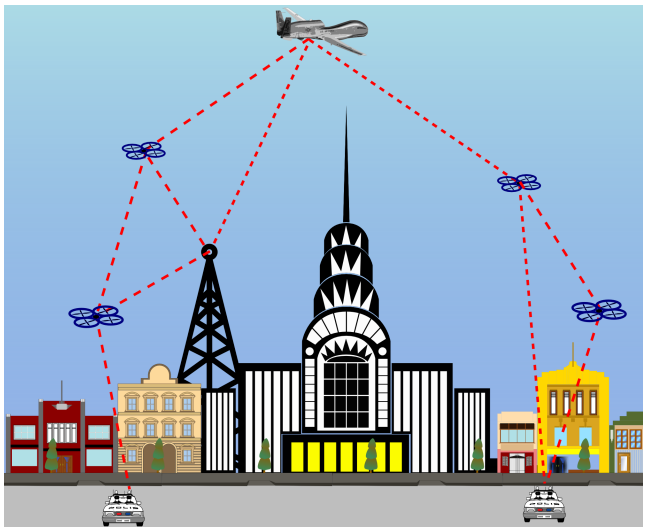


Figure 1: An example of a UAV 3D mesh network in a urban environment with obstacles.

Testing with UAVs is a very costly, time-consuming event which requires a lot of manpower, since failure in communication or any other mechanical malfunction may cause fatal accidents. Also, in a real test environment, there are many parameters involved making the comparison of two different algorithms very difficult to perform. For example, the signal propagation characteristics change throughout the day, or even with the local humidity. In addition to all these factors, flying UAVs in outdoor spaces is prohibited due to the risk of accidents with civilians. However, since 3D mesh network is gaining its importance in scenarios such as first responders in disaster situations or firefighter networks, the

WNS3, June 15-16, 2016, Seattle, WA, USA

DOI: http://dx.doi.org/10.1145/2915371.2915384

required network protocols must be tested intensively before deployment. Simulations are used widely to model the networks that can emulate the behavior of UAVs. Simulation can reproduce the exact same environment when comparing two different protocols. Another huge problem arises when testing the scalability of protocols, i.e. the reaction of a protocol when a larger number of nodes is present in the network. After performing rigorous simulations, a protocol or a model is tested with real hardware. To obtain realistic performance, some networks with mobility should be tested in an environment with large obstacles, the deployment of which can be financially infeasible. Federal Aviation Administration restricts flying and testing with UAVs to mostly rural areas such as in the state of Nevada [1]. Since testing UAV networks in urban areas is not possible, simulation of UAV networks should be done in a reliable network simulator that provides seamless integrity of mobile nodes (MNs) with a wide range of upper layer protocols, and also provides the capability of 3D mobility with obstacles.

An example of 3D UAV mesh can be seen in Figure 1. Different classes of UAVs are depicted at different layers. One hop links between two neighbors fluctuate due to multipath signal propagation through reflection at large objects. The upper layer network protocols are affected by the mobility of MNs as well as the corresponding environment and terrain [15]. For example, in a city with multistoried buildings or in an irregular terrain, the link between MNs may break when a building is between them. Since the application scenarios are not limited, UAV mobility is desired to be integrated with simulator where the application and other protocols are implemented. The movements of MNs are created using standard mobility models. These models are now incorporated into the networks simulator to describe the migration of MNs. The mobility of an MN can be represented in a simulator in two ways: *traces* and *synthetic models*. Traces are used to provide real-life recorded data as a time-series position information. Traces are useful to simulate accurate systems, but in multiple MN scenarios with long time simulation it is space consuming. Synthetic models are widely used in simulators to mimic the movement of MNs movement in MANET for performance evaluation. Mobility models characterize how MNs change their velocity and direction over time. A wide range of mobility models are presented in literature such as *random walk, random waypoint, random direction, Gauss-Markov, column formulation*, and *nomadic community*. A detailed description of the mobility models commonly found in network simulators can be found in [9] where a comprehensive study on the mathematical properties of many models is detailed.

This paper introduces three extended mobility models to allow obstacles in the environment. The models were developed for ns-3 [2] using the existing implementations when possible. We show preliminary network performance results, and also provide free access to the source code[1]. The main contributions of the paper are:

- Description of three-dimensional mobility models
- Algorithms for how the models behave with obstacle avoidance
- Open-source implementations of the models in ns-3

[1]http://www.cse.unr.edu/~regis/ns-3

The remainder of the paper is organized as follows: Section 2 provides an overview of the existing systems and their weaknesses. Sections 3, 4, and 5 discribe the implementation of the new models and features. A brief demonstration and performance comparison of these models is presented in Section 6. Finally Section 7 concludes the paper.

2. MOBILITY MODELS AND SIMULATORS

Mobility model is a set of rules that governs how a mobile node moves. It rules how the velocity, acceleration, and location of a node change over time. These mobility models are necessary for simulation purposes when investigating new communications protocols and techniques. Currently many mobility models have been proposed, but their movements are inspired by specific applications and scenarios.

For example, the Manhattan Mobility model [17] uses a grid-like topology of paths in which a node is constrained to move. This model aims to imitate the mobility of vehicles in a urban city environment, and is particularly useful in simulating vehicular ad hoc networks (VANETs). However, the movements of UAVs are not limited by the street paths of a city. If a node has the ability to fly with high enough altitude, its movements may not be physically limited at all. In these cases, more suitable mobility models should be used to simulate the trajectory. Here is where random models come into the picture, to simulate the movements of a node without the knowledge of its mission or task. The Random Walk mobility model defines a sequence of random steps in which a node traverses a certain area. This model establishes how long and in which direction the node should move before changing its direction once again. The problem with this model is that the node density in the simulation is more concentrated in the center of the scenario and less near the boundaries. The Random Direction mobility model is a little different from the random walk, because it forces the node to travel until it reaches the border before changing direction, not limiting the trajectory by a certain amount of time. Another branch of the random models are the temporal dependency mobility models. A popular example is the Gauss-Markov model, in which the direction and speed of a node depends on its previous direction, thus limiting these parameters within a certain range [9].

Authors in [12] proposed a new MANET mobility model called Realistic Mobility Model, in which the node velocities and directions are based on probability distributions that imitate real user mobility behavior. The results create trajectories that resemble real mobility traces. The authors in [11] presented a new VANET mobility model. The movements are limited to the streets in a city, and the speed of a node changes based on its neighboring nodes, similar to a real situation in the streets. In [13] two different mobility models for common applications of fixed wings aircrafts are proposed. The first is a simple random model with dependency on the last action taken by the aircraft: turn right, turn left or straight ahead. The second is a pheromone repel model, where the probability of taking an action depends on the other aircraft's movement as well. The models however were created to simulate cooperative reconnaissance applications. [8] proposed a mobility model for UAVs based on the most common movement patterns executed by commercialized products such as the paparazzi. The node selects one

of the predefined patterns based on the probability of a pattern being chosen. Despite the complexity and resemblance of the models to the real world, there is a lack of proposed models for three-dimensional scenarios, and even less when it comes to obstacles. In an effort to fill this gap, this paper presents two mobility models that are both obstacle and 3D compliant, imitating the movements of modern UAVs.

Simulation plays an important role when creating new networking techniques and protocols. Before building the hardware and software that implements a new feature, the simulations help to decide the feasibility of the new technology; if it does not perform well, it most likely will not perform well in the real world. The mobility of nodes is an important aspect of a network. It directly influences the topology of the network, the links formation, and link failures. Therefore, it is critical to build reliable simulation models to assist the development of more advanced technologies. A network simulator consists of a series of different models that imitate certain behaviors and protocols, glued together to analyze the performance of the system. There are some network simulators available for use that implement these mobility models. OPNET [5] is a commercial platform specialized in telecommunication transmission products for access networks. OMNeT++ [3] is a simulation framework that relies on different modules to provide functionalities rather than having an extensive core library. Its modules are developed independent of each other, following their own development schedule. ns-3 [2] is a discrete event network simulator, created to overcome the weaknesses of its ancestor ns-2. It is designed to facilitate the creation and integration of new models into the core of the simulator. The project is maintained by a community of researchers affiliated with reliable educational institutions, and main libraries receive constant scheduled updates. It offers a means to integrate third-party tools, such as Simulation of Urban MObility (SUMO) [6]. Since the primary purpose of this simulator is to serve the educational and research communities, its modules and source codes are kept under open-source license. Due to its easy integration and support from the community, ns-3 was chosen as the development platform for the work presented in this paper. ns-3 assumes the Cartesian coordinate system represented by the tuple (x, y, z), while the mobility models use the spherical coordinate system composed by (r, θ, ϕ). In this paper the radius r can be referred to as v when representing speed. Figure 2 shows the coordinate systems.

There are many different applications for a UAV network: surveillance, border control, first response in emergency situations, and so on. The movement patterns of each mission can be very different from the other. In the same network for example, different nodes may perform various tasks, but still serve as relay nodes in the communication network. Current simulators lack the ability to imitate the movements enabled by the new UAV technologies. UAVs can, for example, change rapidly its direction in both two and three dimensions. They can also share the space in civilian environments with obstacles like buildings and houses. This paper aims to fill this gap by providing new mobility models that allow the simulation of obstacles, and implementing it in a reliable network simulator to be used by the research and educational communities.

The set of libraries contained in the core of ns-3 platform provide the means to simulate not only the common Internet

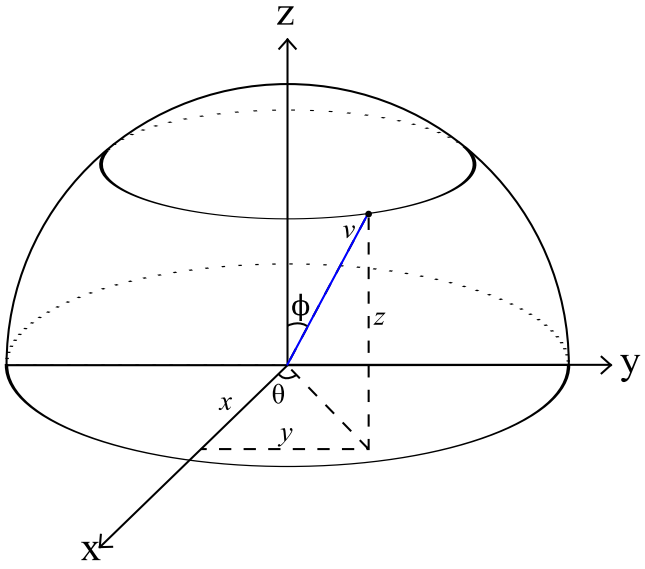


Figure 2: Visualization of coordinates in both cartesian and spherical systems.

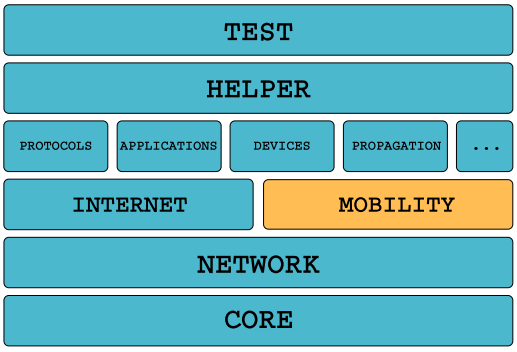


Figure 3: ns-3 block diagram.

but also other networking protocols. The mobility models relevant to this work already implemented and integrated into the ns-3 core libraries are:

- Random Walk (2D)
- Random Direction (2D)
- Gauss-Markov

ns-3 also implements different propagation loss models that accounts for the effect in transmission when buildings are present in the environment. The buildings module provides different physical characteristics of the construction. Commercial buildings, for example, affect differently than residential ones based on their common building materials. This model can be easily reused to construct obstacle compliant mobility models.

An overview of the building blocks and how the simulator works is shown in Figure 3. The core block is where the simulator implements classes with functionalities to facilitate the development of the actual models, like smart-pointers, tracing, logging, and event scheduler. The network

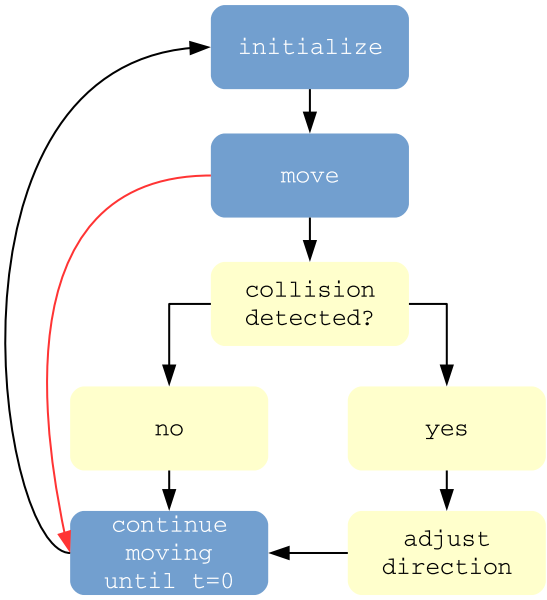


Figure 4: Simplified block diagram of collision avoidance.

block provides abstract base classes for common objects such as packets. These modules are independent of specific networks, and comprise the core that may be used by the upper modules to implement more sophisticated features in a network, not exclusively Internet related. Modules depend only on the modules beneath them. The mobility module, for example, only uses components implemented by the core, while propagation modules depend on the mobility model beneath. The focus of this work is on the mobility module; inside this block resides all the other mobility models implemented in the simulator. Currently the mobility models available to the community do not consider obstacles, they simple assume the node moves in a straight line for a certain period of time.

The flowchart in Figure 4 illustrates where the contributions in the mobility models reside. The adaptations verify if the node collides with the obstacles and adjust the attributes accordingly. Previously the models simply moved from one point to another until the end of the current cycle, denoted by time t in the figure. The new models presented in this paper can be easily used in conjunction with other models to simulate a UAV network. A simple example is the use of SUMO to emulate the movements of ground vehicles while utilizing the 3D models to imitate UAVs, all in the same network. Street maps and building information can be obtained from open sources, such as OpenStreetMap [4], and used with SUMO to generate traces for VANETs [10].

3. RANDOM WALK 3D

Originally the Random Walk model was created to imitate the behavior of particles in physics, but it was later used to simulate movements of nodes in mobile networks. The original Random Walk model implemented in ns-3 defines the movements of a node by the random variables θ, that changes the direction in which the node moves, and speed $v \in [V_{min}, V_{max}]$ contained in a predefined speed interval. This model has two modes: *time mode* and *distance mode*. The first one explicitly decides for how long a node will keep its current speed and direction before choosing new values. The later also decides the time before new values are chosen, but based on the current speed of the node and the predefined *distance* value: $time = distance/speed$. If the mobile node reaches the boundary of the simulation before the resetting process is executed, the node simply bounces back by changing its direction in the corresponding axis and continues moving for the remaining time. For example, if the x-axis is the boundary limit (equivalent to $y = 0$), then the direction in y of the velocity vector is inverted by multiplying it with -1. Similarly, if y-axis is the limit ($x = 0$) the x component of the velocity is inverted.

To enable this model to function in a three-dimensional world, a new random variable is introduced, the aforementioned $\phi \in [0, \pi]$. In addition to the speed and direction, now the node also selects a new pitch at every cycle. Instead of having lines as limits, now the model assumes planes. If the node reaches the limit $xy - plane$ for example, the z component of the velocity vector is inverted to bounce the node back into the scenario $\mathbb{S}$.

Realistic scenarios such as urban environments are not simply empty places where nodes move freely. These scenarios actually introduce certain constraints to the mobility. Even when they move in a 2d plane that is lower than a skyscraper, obstacles must be considered. The implementation of the 3D mobility model in ns-3 provides the feature to simulate movements in the presence of obstacles. The obstacles are assumed to be boxes and axis aligned, meaning their faces are parallel to the axis planes. The effect caused is similar to the bouncing velocity vector: if the node collides with an obstacle it changes the corresponding component of the velocity, depending which side of the object was hit.

In each cycle of the Random Walk model the node chooses new speed, direction, and pitch. These parameters basically define the velocity vector of the node. Then it calculates the next position based on the *time* it is supposed to move with those parameters. If during this period any obstacle is encountered, the orthogonal component of the velocity vector is inverted as mentioned. After the inversion it continues to move with the new values for the remaining time of the cycle, after which a new iteration takes place.

Algorithm 1 shows the pseudo-code of the random walk 3D with obstacles. The implementation required new methods to be added in original classes of the source-code: namely the base classes *ns3::Box* and *ns3::Rectangle*. These modifications are used in all new mobility models. The new method *IsInside(position)* simply verifies if a *position* is inside the object. A method called *WillCollide(position,velocity)* is introduced to the *ns3::Box* class, it verifies if the trajectory of a node will intersect with the box and also returns the intersection point. The new *ns3::RandomWalk3dMobilityModel* class is similar to the original 2d model, except by the new angle variable and the *AddObstacle(obstacle)* method. The *Rebound()* function simply multiplies some of the velocity vector components (*x,y,z*) by -1, depending on which surface of the object collision occurred.

4. RANDOM DIRECTION 3D

In [16] this model was first proposed with the objective of eliminating the concentrated node density problem that occurs in the Random Waypoint Model. In this model the nodes randomly select a point inside the limited area and

Algorithm 1: Random Walk 3D

Input: Boundary ($\mathbb{S}$), set of obstacles ($\mathbb{O}$), default distance ($default\_distance$) or $default\_time$, and Speed limits $[V_{min}, V_{max}]$
Output: List of Time and Position pairs

repeat
  if $mode = time$ **then**
    $default\_distance \leftarrow default\_time * v$;
  $\theta \leftarrow \mathcal{U}(0, 2\pi]$;
  $\phi \leftarrow \mathcal{U}[0, \pi]$;
  $v \leftarrow \mathcal{U}[V_{min}, V_{max}]$;
  $distance \leftarrow default\_distance$;
  $collides \leftarrow false$;
  for $O_i \in \mathbb{O}$ **do**
    if `WillCollide`$(x, y, z, \theta, \phi, O_i)$ **then**
      if $\Delta d >$ `DistToObstacle`$(x, y, z, \theta, \phi, O_i)$ **then**
        $\Delta d \leftarrow$ `DistToObstacle`$(x, y, z, \theta, \phi, O_i)$;
        $collides \leftarrow True$;
  if $collides$ **then**
    $Rebound(\Delta d)$;
  $x \leftarrow x + sin(\theta)cos(\phi)\Delta d$;
  $y \leftarrow y + sin(\theta)sin(\phi)\Delta d$;
  $z \leftarrow z + sin(\phi)\Delta d$;
until $simulation\ ends$;

move there with a certain speed. Since the probability of choosing a point closer to the boundaries is lower than in the middle, the nodes in the scenario eventually end up concentrating in the center. The Random Direction model mitigates this problem by forcing the node to travel until it reaches the boundary of the delimited area.

In the Random Direction model, three variables govern the trajectory of a mobile node: pause P, direction θ, and speed v. Similar to the Random Walk model, these attributes are constrained in their limits. First the node chooses a speed and direction in which it will move. Then it continues moving until it reaches the boundary, standing by that position for P amount of time before reseting the parameters and beginning the cycle once again. Intuitively, the node will only move again if the new direction points towards the middle of the limited area, otherwise it will remain in the paused state.

The movements in Z-axis are allowed by the introduction of the pitch ϕ in the system. In the main loop of the algorithm the model selects the values for the random variables ϕ, θ, and v. Then it calculates the intersection of the new trajectory path of the node with each obstacle inside the scenario, and the scenario boundaries itself. The next position where the node will randomize the values again is the intersection with the smallest distance to travel. If an object is not in the path of the node, the intersection distance returns infinity. After the node reaches the destination, it pauses for a random amount of time, constrained within the interval $P \in [P_{min}, P_{max}]$.

Algorithm 2 shows the pseudo-code of the random direction 3D. The implementation utilizes the same new methods included in base classes *ns3::Box* and *ns3::Rectangle* in the source-code. The class *ns3::RandomWalk3dMobilityModel* also adds a new angle to allow the movement in z-axis. To keep consistent, the same *AddObstacle(obstacle)* method was implemented.

Algorithm 2: Random Direction 3D

Input: Boundary ($\mathbb{S}$), set of obstacles ($\mathbb{O}$), Pause time limits (P_{min}, P_{max}) and Speed limits $[V_{min}, V_{max}]$
Output: List of Time and Position pairs

repeat
  $Pause(\mathcal{U}[P_{min}, P_{max}])$;
  $\theta \leftarrow \mathcal{U}(0, 2\pi]$;
  $\phi \leftarrow \mathcal{U}[0, \pi]$;
  $v \leftarrow \mathcal{U}[V_{min}, V_{max}]$;
  $\Delta d \leftarrow$ `DistanceToBoundary`(x, y, z, θ, ϕ);
  for $O_i \in \mathbb{O}$ **do**
    if `WillCollide`$(x, y, z, \theta, \phi, O_i)$ **then**
      if $\Delta d >$ `DistToObstacle`$(x, y, z, \theta, \phi, O_i)$ **then**
        $\Delta d \leftarrow$ `DistToObstacle`$(x, y, z, \theta, \phi, O_i)$;
  $\Delta t \leftarrow \Delta d / v$;
  $x \leftarrow x + sin(\theta)cos(\phi)v\Delta t$;
  $y \leftarrow y + sin(\theta)sin(\phi)v\Delta t$;
  $z \leftarrow z + sin(\phi)v\Delta t$;
until $simulation\ ends$;

5. GAUSS-MARKOV

The movements of an object in real world may be limited and constrained by the laws of physics. Hence, the change in velocity and direction may depend on the previous values of these attributes. This correlation between the speed/direction in different iteration time slots is not considered when Random Models are used. In random approaches, the nodes' movements and speed may vary abruptly from one instant to another. But when considering real objects like cars, helicopters, and airplanes, even though some may be able to change direction faster than others, these movements are not as quick as the random models imply. The Gauss-Markov mobility model, on the other hand, adds this temporal dependency characteristic when the mobile node is moving around the scenario, enabling the limitations of real world movements to be emulated.

This model assumes the velocity of a mobile node is correlated over time. The changes in velocity are modeled as a Gauss-Markov stochastic process. The new value is constrained within a limited range, with a mean and standard deviation. The model also has a tunning parameter $\alpha \in [0, 1]$, known as the memory level. This parameter dictates the importance of old values when calculating the new ones. This tunning parameter gives the ability to imitate the behavior of other models. For example, if α is low, the velocity is determined by random variables and not the previous values, making the behavior closer to the ones of random models.

The Gauss-Markov model also assumes a bounding box to limit the movements of the mobile node. The velocity and directions of the node are changed after a predefined amount of time. During this time interval the node moves with fixed speed and direction. When it finishes moving for the interval, new values are chosen based on the previous

ones and the system defined parameters. The system parameters define, for example, the deviation the new speed will have from the previous one. If the node moves towards the limits of the simulation, the direction is shifted by 180 degrees (πrad), forcing the node to stay within the boundaries. More details of the original model can be found in [14].

The implementation of obstacles in this model follow the same principle as when the node gets closer to the bounding limits. If the mobile node is moving towards an obstacle, its direction and pitch are shifted, forcing it to bounce back inside the simulation limits. The trajectory appears to be smoother than the random models, giving the simulations a more realistic behavior of a mobile node's movement. Since the 3D Gauss-Markov model is already implemented in ns-3, the only modifications made were to allow the addition of obstacles with *AddObstacle(obstacle)* and the collision verification, which is basically the same as the one to verify if the node is out of bounds, and adjust the new directions accordingly.

6. DEMONSTRATION AND RESULTS

To illustrate the behavior of the enhanced models, a sample trajectory for each of the described methods is shown in Figure 5. In Figures 5a and 5b the Random Walk models are configured in different modes: *time* and *distance* respectively. Figure 5c shows an example of the Random Direction model, while the trajectory in Figure 5d is defined by the Gauss-Markov model. Only one obstacle is placed in this illustration to facilitate the visualization of the sampled trajectory. The object is centered in $(50, 50)$ and has dimensions of $100m$ width and length, and $200m$ height.

The simulated scenario has dimensions of $300 \times 300 \times 200(m)$. The speed of the node is set to an average of $25m/s$, and the simulation runs for $200s$. In the *distance* mode of the Random Walk, the node travels for $200m$ (equivalent to *time* mode with $200/25 = 8s$ interval) before changing direction, while in the *time* mode it travels for $10s$ before resetting.

Table 1: Obstacle dimensions.

Parameter (meters)	Obstacle 1	Obstacle 2
Center coordinates	(50,50)	(150,150)
Width	100	100
Length	100	100
Height	100	200

To provide some insights on the behavior of an ad hoc network, we executed preliminary simulations comparing the performance in two cases: with and without obstacles. When simulating with obstacles, two objects are placed in the simulated scenario. Both obstacles have the characteristics of solid *stone blocks*, and the detailed position and dimensions are depicted in Table 1. We traced the packet delivery ratio of control packets (CPDR) of the Optimized Link State Routing (OLSR) protocol. The general parameters for the simulations are as follows.

By varying the number of nodes (5 to 50), it is possible to see the degradation in the performance when obstacles are considered. For example, the Gauss-Markov model has CPDR of 80.9636% and 83.9625% when simulating 50 nodes with and without obstacles, respectively. The same trend can be observed for the other models as well, and the performance is also influenced by the mobility model utilized.

Table 2: Simulation parameters.

Parameter	Value
No. of nodes	20
Simulation time	200 s
Traffic generator	OnOff Application
Transport protocol	UDP
Routing protocol	OLSR
WiFi standard	IEEE 802.11b
Datarate	1 Mbps
Speed	5 to 25 m/s (25 default)
Propagation loss model	Okumura-Hata

From the graph in Figure 6 we see that Gauss-Markov model results in a deteriorated performance compared to the other random models.

Figure 7 illustrates the performance of the network. The traffic was generated by half of the nodes, the other half acts as the receiver (i.e. there are 10 information flows with distinct sources and destinations). We compare the results in both cases, with and without obstacles. We can see that the number of reflections increases in a linear trend according to the speed (when there is no obstacle, there is no reflection), as it would be expected. A counter-intuitive result is the hop count, which is in general lower when there are obstacles in the scenario. This happens because the nodes are more restricted in their movements, forcing them to stay closer to each other. Hence, if there is no obstacle, they have more room to move, and the probability of augmenting the distance is higher. Following the same logic, if the number of hops increases, the end-to-end delay and jitter also increase if there is no obstacle.

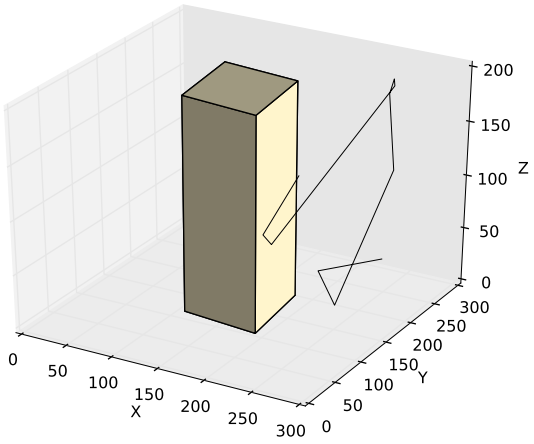
7. CONCLUSION AND FUTURE WORK

In this paper we implement the three different mobility models in a reliable network simulator framework to emulate a three-dimensional world. One requirement for the development of the models is to make them obstacle compliant, to meet the requirement when simulating realistic environments such as urban areas with buildings or mountains. The models are implemented in the network simulator ns-3, due to its open-source nature, and the focus in research and educational communities. The models were demonstrated in a simple scenario, resulting in the expected degradation of performance when the obstacles are present in the simulation.

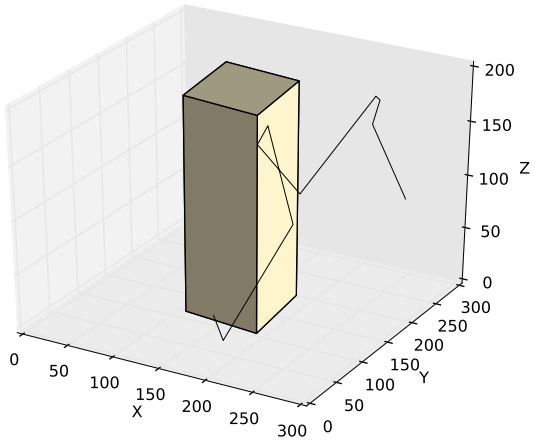
In the future, more complex algorithms for obstacle avoidance can be implemented to help simulate in case the path of the node is known, like in a particular mission known to the system developer. Such algorithms already exist and are constantly being studied by the robotics research community. Integrating these two remains a challenge to be explored. This work provides the means to anyone interested in quickly deploy simulations of UAV networks, without the need of external tools to obtain the movements of the nodes in the network.

8. ACKNOWLEDGMENTS

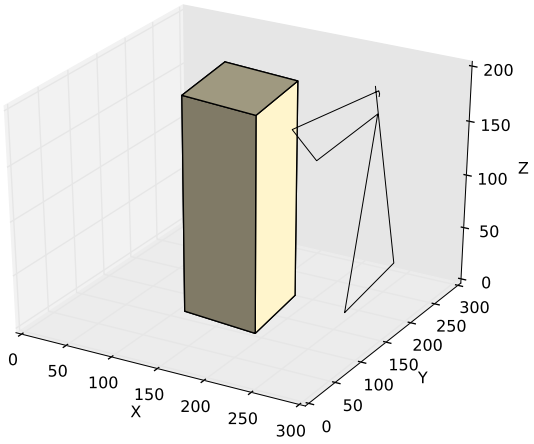
This research was supported by the National Science Foundation, Partnership for Innovation Program, Grant No. 1430328 and CAPES Brazil 13184-13-0.



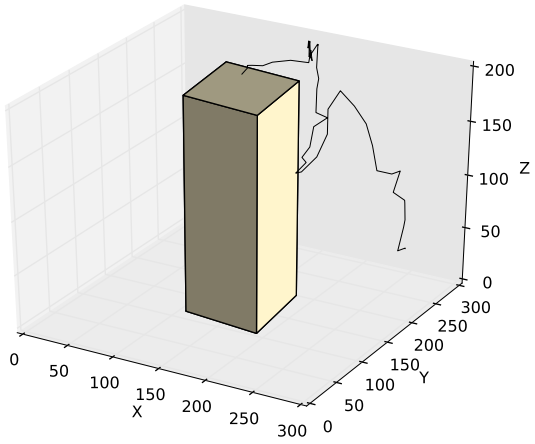
(a) Random Walk (Time)



(b) Random Walk (Distance)

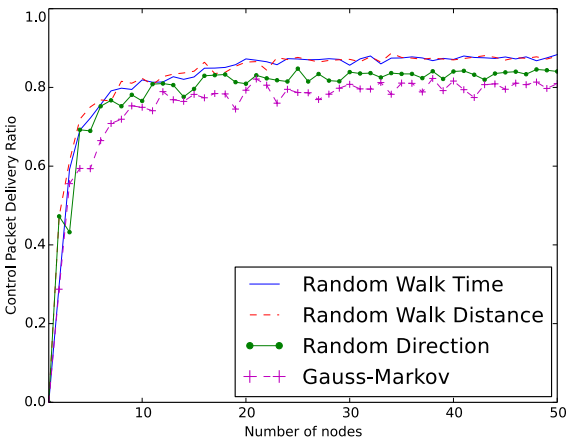


(c) Random Direction

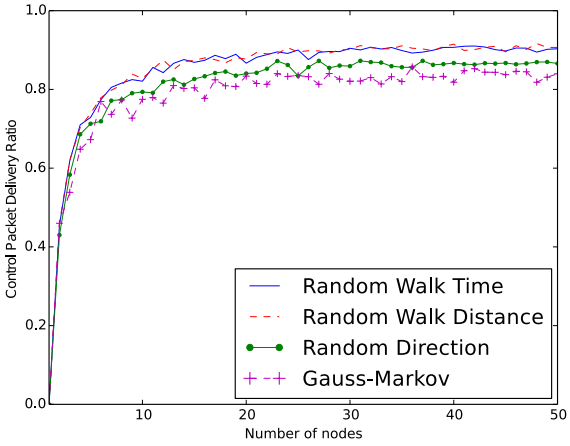


(d) Gauss-Markov

Figure 5: Trajectory with one obstacle.

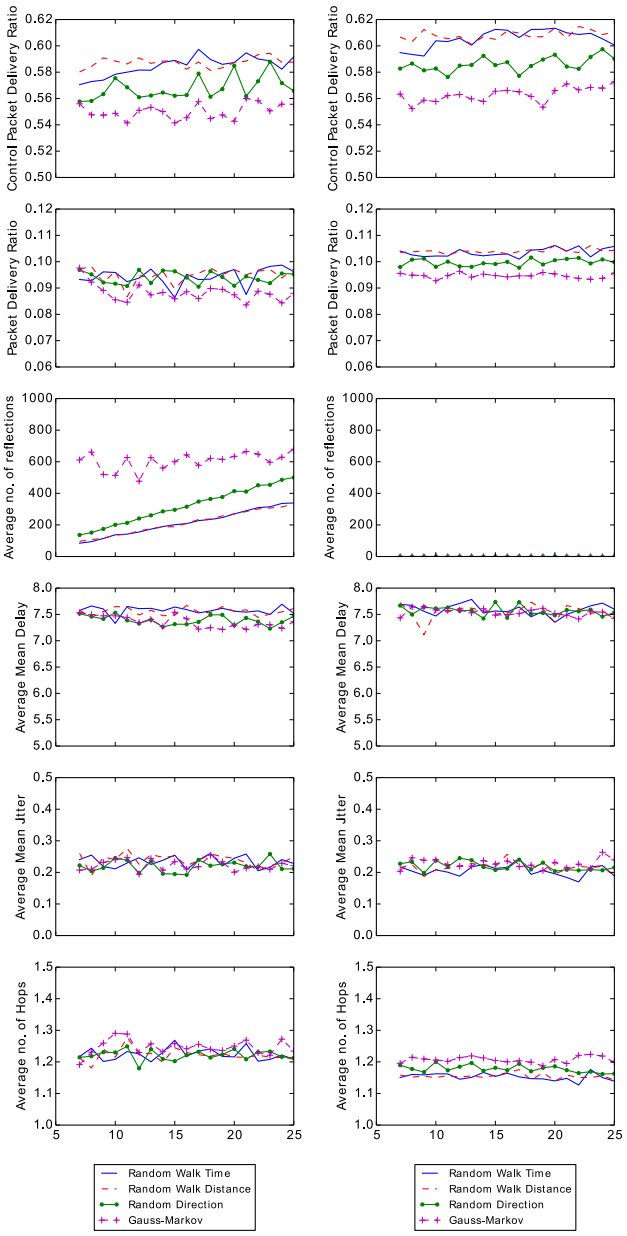


(a) With obstacles



(b) Without obstacles

Figure 6: Control packets delivery ratio vs number of nodes.



(a) With obstacles **(b)** Without obstacles

Figure 7: Network performance parameters.

9. REFERENCES

[1] Federal Aviation Administration approved UAS test sites. https://www.faa.gov/uas/legislative_programs/test_sites.

[2] ns-3. https://www.nsnam.org.

[3] OMNeT++. https://omnetpp.org.

[4] Open Street Map (OSM). https://www.openstreetmap.org.

[5] Opnet. http://www.riverbed.com.

[6] Sumo. http://sumo.dlr.de.

[7] I. Bekmezci, O. K. Sahingoz, and Ş. Temel. Flying ad-hoc networks (fanets): A survey. *Ad Hoc Networks*, 11(3):1254–1270, 2013.

[8] O. Bouachir, A. Abrassart, F. Garcia, and N. Larrieu. A mobility model for uav ad hoc network. *2014 International Conference on Unmanned Aircraft Systems (ICUAS 2014)*, pages 383–388. IEEE, 2014.

[9] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless communications and mobile computing*, 2(5):483–502, 2002.

[10] S. E. Carpenter and M. L. Sichitiu. An obstacle model implementation for evaluating radio shadowing with ns-3. In *Proceedings of the 2015 Workshop on ns-3*, WNS3 '15, pages 17–24, New York, NY, USA, 2015. ACM.

[11] D. S. Gaikwad and M. Zaveri. A novel mobility model for realistic behavior in vehicular ad hoc network. *IEEE 11th International Conference on Computer and Information Technology (CIT 2011)*, pages 597–602. IEEE, 2011.

[12] A. E. Kamal and J. N. Al-Karaki. A new realistic mobility model for mobile ad hoc networks. *IEEE International Conference on Communications (ICC 2007)*, pages 3370–3375. IEEE, 2007.

[13] E. Kuiper and S. Nadjm-Tehrani. Mobility models for uav group reconnaissance applications. *International Conference on Wireless and Mobile Communications (ICWMC 2006)*, pages 33–33, July 2006.

[14] B. Liang and Z. Haas. Predictive distance-based mobility management for pcs networks. In *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1377–1384, March 1999.

[15] D. B. J. D. A. Maltz and J. Broch. DSR: The dynamic source routing protocol for multi-hop wireless ad hoc networks. *Computer Science Department Carnegie Mellon University Pittsburgh, PA*, pages 15213–3891, 2001.

[16] E. Royer, P. Melliar-Smith, and L. Moser. An analysis of the optimum node density for ad hoc mobile networks. *IEEE International Conference on Communications (ICC 2001)*, volume 3, pages 857–861, 2001.

[17] E. SMG. Universal mobile telecommunications system (umts); selection procedures for the choice of radio transmission technologies of the umts. *ETSI Document TR*, 101:112, 1997.