Modeling Communication over Terrain for Realistic Simulation of Outdoor Sensor Network Deployments

SAM MANSFIELD, KERRY VEENSTRA, and KATIA OBRACZKA, University of California at Santa Cruz

14

Popular wireless network simulators have few available propagation models for outdoor Internet of Things applications. Of the available models, only a handful use real terrain data, yet an inaccurate propagation model can skew the results of simulations. In this article, we present TerrainLOS, a low-overhead propagation model for outdoor Internet of Things applications that uses real terrain data to determine whether two nodes can communicate. To the best of our knowledge, TerrainLOS is the first terrain-aware propagation model that specifically targets outdoor IoT deployments and that uses height maps to represent terrain. In addition, we present a new terrain classification method based on terrain "roughness," which allows us to select a variety of terrain samples to demonstrate how TerrainLOS can capture the effects of terrain on communication. We also propose a technique to generate synthetic terrain samples based on "roughness." Furthermore, we implemented TerrainLOS in the COOJA-Contiki network simulation/emulation platform, which targets IoT deployments and uses TerrainLOS to evaluate how often a network is fully connected based on the roughness of terrain, as well as how two popular power-aware routing protocols, RPL and ORPL, perform when terrain is considered.

CCS Concepts: • **Networks** → **Network simulations**; *Network experimentation*; *Network performance modeling*; *Network performance analysis;*

Additional Key Words and Phrases: Modeling communication, terrain, network performance

ACM Reference format:
Sam Mansfield, Kerry Veenstra, and Katia Obraczka. 2022. Modeling Communication over Terrain for Realistic Simulation of Outdoor Sensor Network Deployments. *ACM Trans. Model. Perform. Eval. Comput. Syst.* 6, 4, Article 14 (April 2022), 22 pages.
https://doi.org/10.1145/3510306

1 INTRODUCTION

The advent of the **Internet of Things (IoT)** has renewed interest in wireless sensor networks and their applications, such as environmental and habitat monitoring (e.g., air pollution and detecting forest fires), surveillance of public spaces, and smart agriculture, to name a few. These applications rely on a network of many small, low-power, resource-constrained devices equipped with sensors that are deployed in the region being monitored. Sensor readings are then transmitted through the network to a data collection site, also known as the data sink, to be processed locally or remotely (e.g., in the cloud).

Authors' address: S. Mansfield, K. Veenstra, and K. Obraczka, University of California at Santa Cruz, Department of Computer Engineering, 1156 High St. Santa Cruz, CA, 95064, USA; emails: {smansfie, veenstra}@ucsc.edu, katia@soe.ucsc.edu.

2376-3639/2022/04-ART14
https://doi.org/10.1145/3510306

Wireless sensor networks have attracted considerable attention from the network research community and have been the subject of a vast body of work. Network simulation platforms have been particularly useful in the development, testing, and evaluation of wireless sensor networks and their protocols. However, studies have questioned the fidelity of simulation platforms [15, 25, 32]. One of the main culprits is the channel propagation modeling used by network simulators in general, and sensor network simulators in particular, which tend to be overly simplistic and do not accurately simulate reality. For instance, the work in Reference [20] reviewed MobiCom and MobiHoc publications from 1995 to 2003 and reported that the use of overly simplistic propagation models outweighed accurate models. The study also noted that routing protocols perform differently depending on the propagation model used. Consequently, even if a study is only comparing two different algorithms, the propagation model itself can determine the outcome. Although this study is somewhat dated in 2020, its point still stands today. This realization and our own experience using network simulation platforms motivated us to explore a topography-aware propagation model that, while computationally simple, is able to more accurately simulate outdoor sensor networks.

As a result, we introduce a new *terrain-aware* propagation model that we call TerrainLOS, which stands for communication over Terrain using Line Of Sight, designed specifically for outdoor IoT scenarios. Our model aims to strike a balance between complexity and accuracy when simulating outdoor IoT deployments. TerrainLOS, which is described in detail in Section 3, uses **Digital Elevation Models (DEMs)** [8], which are digital representations of a terrain's surface, to determine whether wireless nodes have line-of-sight communication at typical IoT frequency bands (i.e., 2.4 GHz and 900 MHz). Besides being computer-friendly terrain representations, DEMs are publicly available for numerous geographical regions around the globe. TerrainLOS is designed and optimized specifically to be used as a propagation model to simulate outdoor IoT deployments by using DEMs to account for the effect of terrain when establishing line-of-sight communication between nodes. This yields improved simulation fidelity and allows IoT simulation experiments using TerrainLOS to be used to complement real testbed experiments when evaluating and validating IoT protocols. Additionally, TerrainLOS is computationally simple and does not significantly impact simulation time. In our work, we use TerrainLOS to show how different types of terrain affect the performance of IoT networks and their protocols.

A preliminary version of our work appeared in Reference [18]. This article extends our previous work and provides the following contributions:

(1) We survey terrain-aware propagation models that have been implemented in well-known network simulators and discuss how they differ from our own work. More specifically, we provide a detailed description of the approach used by TerrainLOS to determine whether two nodes can communicate over terrain and why we chose this method over existing ones.
(2) We analyze TerrainLOS' memory footprint and runtime as a function of the area under consideration and number of nodes, respectively, and show that it does not significantly increase simulation time.
(3) We introduce a new way to classify terrain based on terrain "roughness" that extends the metric used by Veenstra and Obraczka [30]. We call the proposed metric **Average Cumulative Visibility (ACV),** which measures the visibility of a terrain as it affects line-of-sight communication.
(4) We propose a novel approach to modeling terrain roughness using Cumulative Visibility distributions. Using the proposed terrain modeling approach, terrain maps can be generated with the desired roughness instead of having to manually search for terrain areas that match the desired roughness. To validate the proposed terrain generation method, we compare the

connectivity of a network using modeled terrain against an actual terrain with the same roughness.

(5) We demonstrate TerrainLOS to show how terrain can affect the performance of networks and their protocols. We implement TerrainLOS in the COOJA-Contiki sensor network simulation/emulation platform and evaluate two well-known IoT routing protocols, namely RPL and ORPL. We run experiments based on what we originally reported [18], but more scrutiny is placed on whether the network is connected, the average degree of the network, and using different IoT deployments/topologies. We also run additional experiments to observe the connectivity of a network based on the underlying terrain's ACV.

To the best of our knowledge, TerrainLOS is the first terrain-aware propagation model that specifically targets outdoor IoT deployments and uses height maps to represent terrain. The rest of the article is organized as follows. In Section 2, we provide an overview of related work that focuses on developing and implementing terrain-aware propagation models for network simulation and emulation platforms and how these efforts differ from our own. In Section 3, we present the TerrainLOS terrain-aware propagation model; we also evaluate the TerrainLOS storage and computation footprint based on its implementation in the COOJA-Contiki network simulator/emulator platform. We then discuss our terrain classification approach in Section 4. In Section 5, we present experimental network connectivity results over various classes of terrain based on our classification scheme. We then discuss a way to create modeled terrain given an ACV in Section 6, as well as a comparison of connectivity over modeled terrain against experimental connectivity results presented in Section 5. In Section 7, we present experimental results showing the impact of terrain on well-known IoT routing protocols. We discuss concluding remarks and directions for future work in Section 8.

2 RELATED WORK

There is a vast body of work on wireless channel propagation models. For instance, considerable attention has focused on using empirical data collected in the field, e.g., signal strength over distance, to model signal propagation. Alternatively, there are approaches that use real-world (or sometimes, synthetically generated) height data to measure connectivity. The differences between these two approaches may seem subtle, but the key distinction is that in the case of approaches that use empirical data (e.g., signal data measured in the field), line-of-sight is implicitly accounted for, while in approaches that use height data, line-of-sight connectivity needs to be computed.

In this section, we provide an overview of existing approaches that fall under the height data modeling category, since they are the closest to TerrainLOS. In particular, we review related work on propagation models implemented in network simulators and that use terrain height maps to determine connectivity.

From the literature, we have found five network simulators that implement terrain-aware propagation models, namely `ns-2` [1], `ns-3` [22], OMNeT++ [2], OPNET (now known as Riverbed Modeler) [3], and COOJA [24]. Most implementations are based on Durkin's model [13], which uses line-of-sight and Fresnel Zone clearance to determine connectivity. Line-of-sight calculations vary slightly by implementation but are similar to TerrainLOS. Fresnel Zone clearance is based upon the idea that two radios, even with line-of-sight clearance, will experience signal strength reduction based on the obstruction of the ellipse between them. As described in more detail in Section 3, TerrainLOS does not take into account Fresnel Zone clearance as the accuracy of typical height data, e.g., in DEM repositories is below the Fresnel Zone ellipsoid's small radius at typical IoT frequencies (900 MHz and 2.4 GHz) and communication ranges.

In Reference [14], Durkin's model was implemented in ns-2 [1], and coverage was evaluated against the Longeley–Rice model (also known as the Irregular Terrain Model) on several terrain maps and was shown to be within 5% coverage difference. The speed of simulation and the throughput were measured by length of simulation using one terrain map and generated throughputs from 0.1–7 Mbps using a fixed transmission range of 150 m with and without node mobility. At low data rates the simulation time overhead was close to zero when using the implemented model, but at 7 Mbps considering terrain and using stationary nodes, the simulation time overhead was twice as long (14 hours vs. 7 hours) as using a model that did not consider terrain. Node transmission range was fixed at 150 m, and connectivity of the network was not reported. This work also evaluated throughput performance using five generated terrain types, namely flat, hillside, hill, ravine, and pyramid. The authors showed that terrain can both negatively and positively impact network performance.

In our work, we go a step further by evaluating throughput, latency, and energy consumption of power-aware routing protocols under different deployments using a variety of terrain according to our roughness-based classification. For a fair comparison, we isolate the effect of terrain by ensuring that only connected networks are considered in our simulation experiments. In addition, we evaluate TerrainLOS in terms of its computation overhead and memory footprint.

The Durkin's model was ported to ns-3 [22] in the work reported in Reference [28]. The overhead of the model was evaluated as a function of the terrain map's resolution. As expected simulation time increased for higher resolution maps. In our work, we show how TerrainLOS affects computation time and memory overhead based on number of nodes and terrain size. In addition, we propose a way to classify terrain based on "roughness" and introduce a terrain model that can be used to generate terrain of different "roughness." We also run simulation experiments to compare throughput, latency, and energy consumption to evaluate the performance of power-aware routing in a variety of IoT scenarios.

In Reference [21], Durkin's model was implemented in the OMNeT++ simulator [2] and simulation startup time overhead, i.e., the time to setup the model at the beginning of the simulation, which involves loading the height data and calculating which nodes have communication line-of-sight with each other, was evaluated as a function of the number of nodes for two pre-selected terrains. As expected, terrain negatively impacted network connectivity, which causes a decrease in simulation time as there are fewer simulation events. In our work, we also find similar results, and for this reason, we make sure we use connected networks to isolate the effects of terrain when evaluating IoT routing protocols.

The OPNET simulator [3] (now known as Riverbed Modeler) has a Terrain Modeling Module that includes different terrain-aware propagation models such as TIREM, Longeley–Rice, and Walfish–Ikegami [14]. TIREM is a closed, proprietary propagation model, so it is not known how it models terrain. The latter two models belong to terrain models that fall under the empirical data class and thus are not comparable to TerrainLOS, which is a height data–based approach.

In summary, TerrainLOS has been designed specifically to simulate outdoor IoT deployments. It is simple, yet accurate given the accuracy of available height maps as well as typical IoT communication ranges and frequency bands. As such, we view TerrainLOS as complementary to existing terrain-aware propagation models, as it focuses on outdoor IoT deployments. For instance, it could be added to the Riverbed Modeler's Terrain Modeling Module as an additional terrain-aware propagation model that can be selected when simulating outdoor IoT scenarios. In this article, we also show that TerrainLOS is lightweight in terms of its computation and memory footprint. Additionally, we provide a way to classify terrain, which we use to evaluate connectivity over different classes of terrain. We also demonstrate the impact of different types of terrain on two well-known IoT routing protocols.

3 TERRAINLOS

The overall goal with TerrainLOS is to compute a graph $G(V, E)$ corresponding to the topology of the network, where V is the set of nodes and E the set of edges or links connecting the nodes, such that there exists an edge $e_{i,j}$ between two nodes i and j if i and j are within transmission range and have line-of-sight with each other. The graph is built at the beginning of a simulation experiment and is consulted to check whether a node can communicate with another node. Line-of-sight connectivity between nodes is determined using Wang's Viewshed Algorithm [31], which is described in Section 3.1.

Typically a terrain-aware propagation model should also check for the first Fresnel Zone clearance. The Fresnel Zone is a region contained by an ellipsoid between two nodes in line-of-sight of each other. If a certain percentage of the Fresnel Zone is blocked, even though the two nodes have line-of-sight, then they will not be able to communicate. More specifically, the radius of the first Fresnel Zone at the widest point can be calculated based on the following formula:

$$r(m) = 8.656\sqrt{\frac{D}{f}},$$

where D is the distance between the two nodes in km and f is the frequency in GHz. If we use 300 m as an optimistically large transmission range distance and 0.9 GHz for the lowest transmission frequency, then we can obtain an optimistic estimate of the Fresnel Zone clearance radius of approximately 5 m. **Shuttle Radar Topography Mission (SRTM)** Digital Elevation Models, which are commonly used because there is a large repository of height data that is publicly available, report a vertical accuracy of 10 m [5], making the 5 m Fresnel Zone radius below the accuracy of the height data we are using. For this reason, we have chosen not to include the Fresnel Zone calculations in our model to save on computational overhead. In other words, for the specific use case of simulating outdoor IoT deployments at typical IoT communication ranges and frequency bands, and using DEMs with resolutions below 5 m, a simplified model without Fresnel Zone clearance provides adequate accuracy.

3.1 Wang Viewshed Algorithm

To determine whether two nodes are in line-of-sight of one another, we use Wang's Viewshed Algorithm [31], which calculates all locations visible from a given point or the point's *viewshed*.

The viewshed of a location, also known as the viewpoint, is the area visible from that given location. Viewsheds can be used for such applications as selecting the location of fire monitoring towers or TV/radio transmission towers, selecting locations with maximum visibility for military surveillance, and selecting locations with minimum visibility for military maneuvers [31]. We use viewsheds to determine locations where communication with the viewpoint over terrain is possible. We implemented Wang's Viewshed Algorithm [31], which was shown to use fewer computations when compared to existing viewshed approaches. As previously discussed, TerrainLOS uses DEMs, or height maps, to represent terrain. DEMs, which are a common way to "digitize" terrain, are essentially two-dimensional matrices where each coordinate represents a geographical location and each value is the elevation at that coordinate, or

$$DEM(geographic\ location) = elevation.$$

For example, the SRTM [27], a joint venture between the National Aeronautics and Space Administration and the National Geospace Intelligence Agency, collects and hosts a large collection of DEMs. SRTM's mission statement is to "topographically map 80% of the earth's surface every 1″ [1 second of arc] with accuracy of 16 m, with 90% confidence."

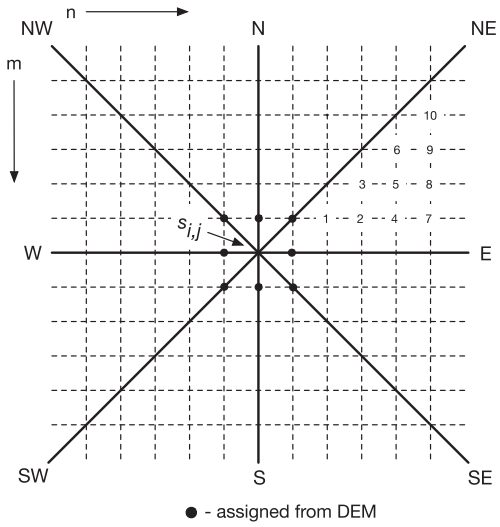

Fig. 1. Viewshed coordinate grid.

Wang's Viewshed Algorithm uses "reference planes" to determine the viewshed from a given viewpoint [31]. A reference plane is the plane that passes through the viewpoint and two adjacent points in the auxiliary grid to the destination, $d_{m,n}$, where $m \in M$, $n \in N$, M is the number of rows of the DEM and N is the number of columns. The auxiliary grid stores the visible height, $r_{m,n}$, from the viewpoint at the location (m, n).

A destination, $d_{m,n}$, is visible from the source viewpoint, $s_{i,j}$, if the destination is on or above the reference plane, $p_{m,n}$, formed using the viewpoint and two adjacent points to $d_{m,n}$ previously calculated in the auxiliary grid. If the destination is visible, then $r_{m,n} = d_{m,n}$; otherwise, the visible height is set to the height at the corresponding location (m, n) that lies on the reference plane. By starting from the source viewpoint and working outward, the auxiliary grid calculations are able to use the previous calculations closer to the viewpoint to create a reference plane.

As depicted in Figure 1, it is simpler to think of the DEM and auxiliary grid as eight sectors and to treat the viewpoint, $s_{i,j}$ as the coordinate origin. The order in which the auxiliary grid is populated varies by sector. The points immediately surrounding $s_{i,j}$ are used directly from the DEM. The vertical, horizontal, and diagonal directions are calculated first moving outward from the viewpoint, as their geometry is simpler.

As an example, to check whether $d_{m,n}$ is visible from $s_{i,j}$ in the East direction, we calculate Z, where Z is the minimum height to make $d_{m,n}$ visible. To calculate Z, the reference plane becomes a reference line as $m = i$ along the east direction. We form the reference line using $s_{i,j}$ and $r_{m,n-1}$,

$$Z = r_{m,n-1}(n - j)/(n - 1 - j).$$

The value $r_{m,n}$ is assigned based on whether $d_{m,n}$ is visible to $s_{i,j}$, i.e., if $d_{m,n} > Z$. If so, then $r_{m,n} = d_{m,n}$; otherwise, $r_{m,n} = Z$.

The remaining points to be calculated are computed based on the sector they lie in. For example, the E-NE sector must calculate points first columnwise outward from the viewpoint and then row-wise outward from the viewpoint; this is also labelled visually in Figure 1.

Keeping with the example of the E-NE sector, the reference plane is formed using the points $s_{i,j}$, $r_{m,n-1}$, and $r_{m+1,n-1}$,

$$Z = r_{m+1,n-1}(-(m - n)/(n - j - 1)) + r_{m,n-1}((n - j + m - i)/(n - j - 1)).$$

The value $r_{m,n}$ is assigned based on whether $d_{m,n}$ is visible to $s_{i,j}$, i.e., if $d_{m,n} > Z$. If so, then $r_{m,n} = d_{m,n}$; otherwise, $r_{m,n} = Z$.

3.2 The COOJA-Contiki Network Simulator/Emulator

We have implemented the TerrainLOS framework inside COOJA-Contiki [24], a well-known network simulator/emulator platform used to evaluate protocols and applications for networks of power-, computation-, and memory-constrained devices, e.g., wireless sensor networks and IoT. While TerrainLOS can be incorporated into any network simulator/emulator that supports radio propagation models, we chose COOJA, as TerrainLOS is designed for the specific use case of IoT deployments, which is the types of networks COOJA targets. Furthermore, to our knowledge, there are no other outdoor propagation models that use real terrain data implemented in COOJA.

As evidence that COOJA is widely used today in the context of IoT and sensor network research, the top five results (from a total of 493) of a Google Scholar search on November 13, 2018 for the keyword COOJA filtered by papers published in 2018 include publications on the following topics: quality of service measurement of a low-power routing protocol [9], software-based energy estimator for localization [17], extending COOJA to include real weather and soil data for agricultural IoT applications [7], connecting another popular network simulator `ns-3` with COOJA [23], and an SDN implementation in COOJA [26].

COOJA simulates low capability nodes, typically called *motes* running the Contiki Operating System [11]. Contiki is a lightweight open source operating system designed for IoT devices that are energy and resource limited. COOJA can emulate the hardware of a variety of IoT devices in addition to the network stack. An advantage of COOJA simulating Contiki OS is that code developed and tested in COOJA can be deployed directly to Contiki motes.

TerrainLOS is currently hosted on Github, and its repository includes installation and configuration scripts for COOJA.

3.3 TerrainLOS Computation and Memory Footprint

In this section, we discuss TerrainLOS's computation and memory overhead. The purpose of this section is to demonstrate that our implementation of TerrainLOS in COOJA minimally affects simulation time and memory footprint. TerrainLOS' overhead is a function of the number of network nodes as well as the size of the terrain, which can also be thought of as the number of digital elevations used. As described in Section 3.1, calculating the viewshed for one node involves examining each digital elevation on the map. To generate a network graph, the viewshed for each node must be calculated.

We implemented Wang's Viewshed Algorithm in the COOJA network simulation/emulation platform [24] by loading the DEM corresponding to the deployment region and calculating the viewshed for each node. Once the viewshed is calculated, each node that is both in transmission range and has line-of-sight is added to the network graph. This is done iteratively, so only one viewshed map is stored at any instance in time. This process happens once at the very beginning of the simulation. The network graph will be re-calculated if the transmission distances change or a node changes location. For all overhead calculations all transmission distances and node locations are fixed. The experimental setup we used to evaluate TerrainLOS' overhead is summarized in Table 1.

When evaluating TerrainLOS' overhead, we consider both memory and computation. TerrainLOS's memory overhead is the memory footprint of loading the digital elevation map as well as the memory used to store the viewshed. Each digital elevation of the map and the viewshed is currently being stored as a Java double, which is 8 B. What we expect is a linear relationship between the number of digital elevations used and the memory footprint.

Table 1. Experimental Parameters for TerrainLOS Overhead Evaluation

Operating System	Ubuntu 12.04 LTS 32 bit virtual machine
Memory	512 MB
Processors	1
CPU	1.8-GHz Intel Core i7
Negotiated Link Speed SSD	3 Gigabit
Virtualization Software	VirtualBox 5.0.26 r108824
Host Operating System	OSX 10.11.15
Computer Model	MacBookAir4,1

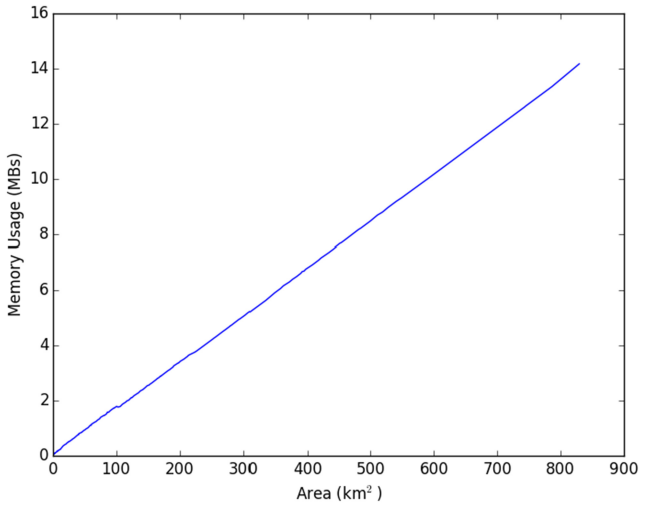

Fig. 2. TerrainLOS's memory overhead.

As discussed in Section 3.1, each digital elevation that we use for this article is one square *arcsecond*. From Figure 2, we observe that, as the number of digital elevations, or the area of the map, increases, the memory footprint also increases.

We measure the memory footprint by using the Java runtime object and measure the difference in memory usage before and after creating the TerrainLOS class. The results are as we expect. As an example, if we look at an area of 200,000 square arcseconds, or 180 km^2, then we would expect to get a memory usage of 200,000*8 B (1.5 MB) for the height map and 200,000*8 B (1.5 MB) for the LOS map, which results in 3 MB. This is almost exactly what we see from the graph in Figure 2. To put these results in perspective, the largest map we used in our experiments contains 10,000 digital elevations (10,000 square arcseconds or 9 km^2). This results in a memory overhead of around 250 KB.

To evaluate TerrainLOS's computation overhead, we measure TerrainLOS's execution time. More specifically, we measure the time from initialization of the TerrainLOS map to finishing the viewshed algorithm computation for every node in the network. We measure the overhead of the viewshed algorithm using the Java Date object. We use three different areas, 10,000 (9 km^2), 50,000 (45 km^2), and 1,000,000 (90 km^2) digital elevations and varied the number of nodes from 1 to 901 by increments of 100. The results are plotted in Figure 3.

As expected, we observe that, given a fixed area, as the number of nodes increases, the computation time also linearly increases. The slope of each line corresponds to the size of the area. As we increase the area, each node has to perform more computations to calculate the LOS map.

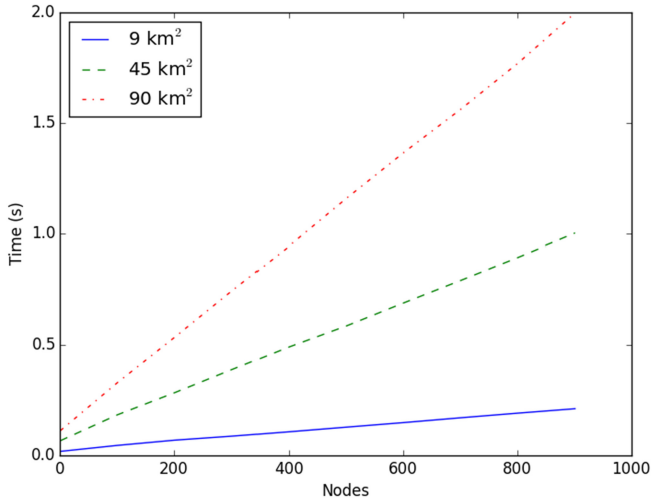

Fig. 3. TerrainLOS's computation overhead.

More concretely, the largest simulation we used in our experiments contained 10,000 digital elevations and 100 nodes. From the graph, we observe that this corresponds to approximately tens of milliseconds, which is almost negligible. In fact, when evaluating COOJA simulation runtimes using 10,000 digital elevation maps, and comparing TerrainLOS against the standard **Unit Disk Graph Model (UDGM)** propagation model, which approximates communication based on whether two nodes are within a certain distance from each other, often simulations using TerrainLOS would actually complete faster than the same simulations running UDGM, as the variance in execution time was a larger factor than the overhead of using TerrainLOS.

4 CLASSIFYING TERRAIN

To evaluate the effect of terrain on networks and their protocols in a thorough and systematic way, it is critical to conduct experiments over a variety of terrain. To this end, we introduce a terrain classification scheme based on the overall visibility of a terrain map.

By classifying terrain, we can run experiments over a range of terrain types and capture how different terrain affect network protocol performance. For instance, results obtained when using flat terrain should match results using a simple disk model. However, experiments over "hilly" regions should exhibit very different behavior. In the remainder of this section, we describe our terrain classification scheme. We start by briefly discussing existing work on terrain classification and modeling to put our work in perspective.

The work in Reference [14], which evaluated the Durkin model in simulation, uses two existing terrain generation tools, namely Terraform [6] and Terragen [4]. They can create terrain based on a given average height and standard deviation. The authors then present a new way to generate terrain using triangular planes.

In our work, we show that terrain can be more accurately modeled using a double Gaussian distribution instead of a height mean and standard deviation. Designing terrain using explicit triangular planes is effective for simple maps such as a hill or a valley but becomes much more computation intensive when handling larger, more complex terrain. As described in Section 6, using our statistical Gaussian method to generate terrain is much more cost-effective for larger terrains with many features.

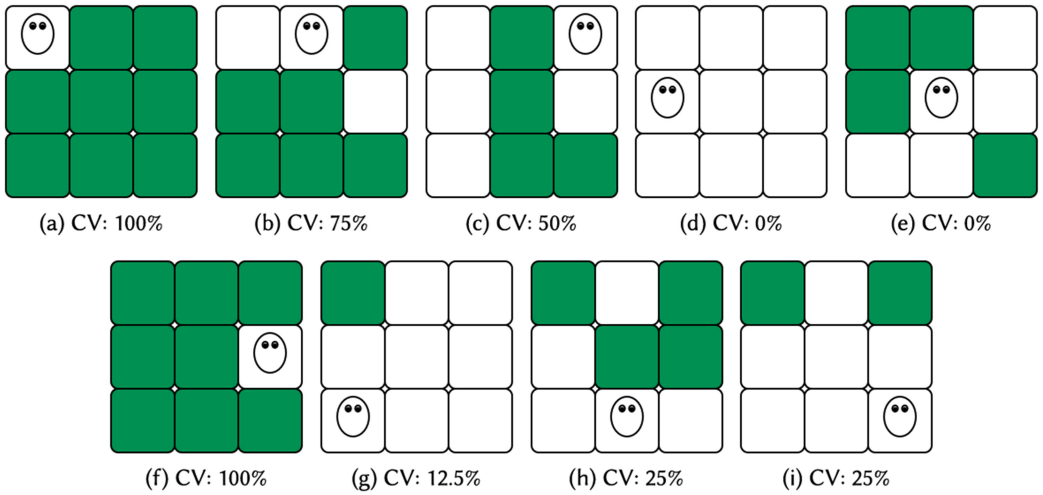

Fig. 4. Pictorial representation of Cumulative Visibility and ACV. The ACV of this map is 48.6%

In Reference [16], the Erdos–Renyi random graph model is used to estimate the probability of connectedness on three selected terrain maps. The authors base these calculations on the probability that two nodes will have line-of-sight. In our earlier work, we independently followed similar rationale and strategy but found this model to be too simplistic. We then propose a double Gaussian distribution–based model (see Section 6) that we validate against actual terrain maps.

We model terrain using a double Gaussian distribution, as described in Section 6, but we classify terrain using the ACV of a terrain area. So, given an ACV, we can generate a modeled terrain that has a given connectivity distribution based on our double Gaussian model. The ACV is based on the *Cumulative Visibility* as defined in Reference [30]. Cumulative Visibility is how many locations are visible on a map from a singular location. We then define the ACV of a terrain map as the average Cumulative Visibility over all the locations in the map. The ACV is then used to classify a map.

Figure 4 provides a pictorial representation of Cumulative Visibility and Average Cumulative Visibility. The face in each image represents the viewpoint and a green square represents a visible location from the viewpoint. The Average Cumulative Visibility is the average of the Cumulative Visibility over all nine locations. In this case, we would get an ACV of 48.6%.

5 CONNECTIVITY OVER TERRAIN

In this section, we explore network connectivity when considering terrain. This is motivated by our performance study (reported in Section 7) comparing two well-known IoT routing protocols, since it is important to be able to control/isolate the effect of network connectivity on network performance.

We consider a network to be *connected* if every node can communicate with every other node on the map through one or more hops. We then evaluate connectivity of a network as a function of the terrain's ACV using TerrainLOS as the underlying propagation model.

Network connectivity is also a function of deployment density, i.e., the number of nodes in the area under consideration. In our work, we define *density* as the ratio between the number of nodes and the number of digital elevation locations in a given map. For example a density of 1% would mean there is 1 node for every 100 digital elevation locations of the map. A density of 100% would mean there are 100 nodes for every 100 digital elevation locations, i.e., there is 1 node for every

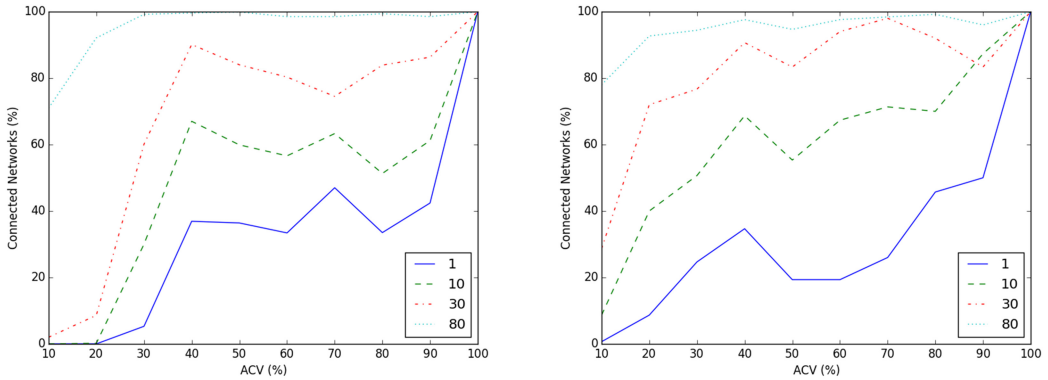

Fig. 5. Connectivity graphs.

digital elevation location. Our method to adjust density was to fix the number of nodes and adjust the map size, i.e., the number of digital elevation locations.

We collected data on at least 150 runs of uniformly random 100 node layouts per ACV and density combination. Terrain was selected by pre-calculating the ACV for a large number of fixed size terrain areas. Then, based on the area we are using, we search for the desired ACV map and choose a random selection. Density was adjusted by fixing the number of nodes at 100 and decreasing or increasing the size of the simulated area. All simulations are run in COOJA using TerrainLOS and are given enough time to run for a graph of the network to be generated, which is then examined to determine connectivity. We report the number of connected networks, i.e., what percentage of runs generated connected networks (i.e., all nodes can communicate with all other nodes).

In these experiments, we started with a fixed transmission range of roughly 150–250 m. However, they resulted in 0% connected networks when the underlying terrain ACV was less than 90%. What we chose to do instead was to set the transmission range to the length of the diagonal of the map, thus making every node within transmission range of every other node. The reason we set the nodes' transmission range to the diagonal of the simulated map is to ensure the network, without accounting for terrain, would be guaranteed to have 100% connectivity, and then when we incorporate terrain we are isolating the effects of terrain on network connectivity, i.e., terrain as the cause for the network not to be connected. While setting the transmission range this high may not practical for real-world applications, as shown in Figure 5(b), we still found that the percentage of connected networks is relatively low. The different curves in Figure 5(b) correspond to the different densities used and, as expected, at lower densities we observe a lower number of connected networks.

6 MODELING TERRAIN

In Section 5, we pre-calculated a large number of terrain areas to generate a variety of maps with a wide range of ACVs. However, this method is very time-consuming, which motivated us to investigate more efficient techniques to build synthetic terrain maps based on the ACV.

As discussed in Section 4, the ACV of a terrain is the average percentage of locations visible across the terrain. For instance, a terrain classified as having an ACV of 10% would mean that on average each location has visibility of 10% of the terrain. As such, a simple way to create a terrain model would be to use the ACV as the probability any two locations on the map are visible from

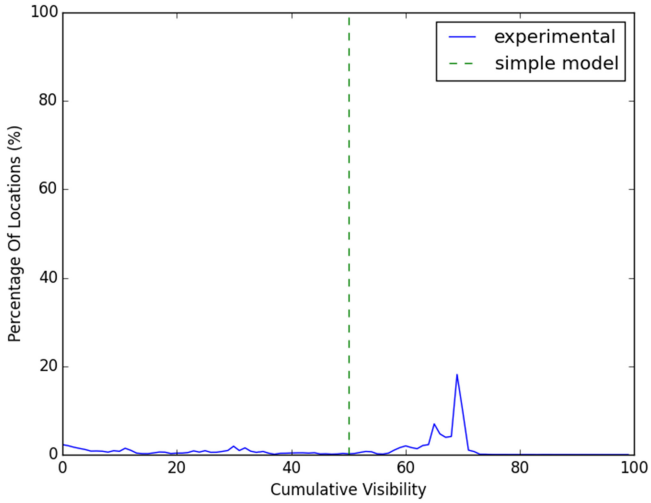

Fig. 6. Simple model compared to experimental visibility distribution at 50% ACV.

one another. We can then use this simple model to build a terrain map by iterating through all possible links, for example, generating a uniformly distributed random number and checking if this random number is less than the ACV, the link is created and the two locations are visible from one another. Note that we assume that links are symmetric, which is consistent with Wang's Viewshed Algorithm; this means that only one chance is given to create a link between any two locations.

The main problem with the model described above is that it is too simplistic and does not resemble reality. More specifically, it tends to predict higher connectedness. The rationale for this is illustrated in Figure 6, which compares the simple model's distribution of Cumulative Visibility against the observed Cumulative Visibility distribution of a 50% ACV map with 100×100 elevations. The latter is obtained by running the Wang Viewshed Algorithm on the average of three 50% ACV maps and recording the Cumulative Visibility at each map location. We then plot the Cumulative Visibility distribution, where each data point represents the percentage of locations with a given Cumulative Visibility of the map.

This simple model uses the ACV to represent the overall visibility of a terrain, which visually results in the "lollipop" seen in Figure 6. But we observe from experimentation that an area with a given ACV exhibits a certain Cumulative Visibility distribution whose "shape" depends on the ACV. When the Cumulative Visibility distribution is averaged across a map, it results in the ACV of the map. However, as shown in Figure 6, representing an area's terrain solely by its ACV results in a model that is too simplistic, as it leaves out information about important terrain features.

By observing Cumulative Visibility distributions of a large number of terrains with different ACVs, we noticed that the distributions exhibit two peaks. One peak is fixed around 0% to 10% locations visible, and the other peak moves depending on ACV. What this tells us is that regardless of ACV, there is always a bit of roughness on a terrain, which as we will see can affect the connectedness of a network. A pictorial representation of the Cumulative Visibility distributions for various ACVs are plotted in Figure 7.

By modeling the observed Cumulative Visibility distribution as two peaks in a two Gaussian model provides considerably more fidelity to reality. The peak from 0% to 10%, which we call the *small peak*, is fixed across ACV and the other peak, the moving peak, which we can call the *large*

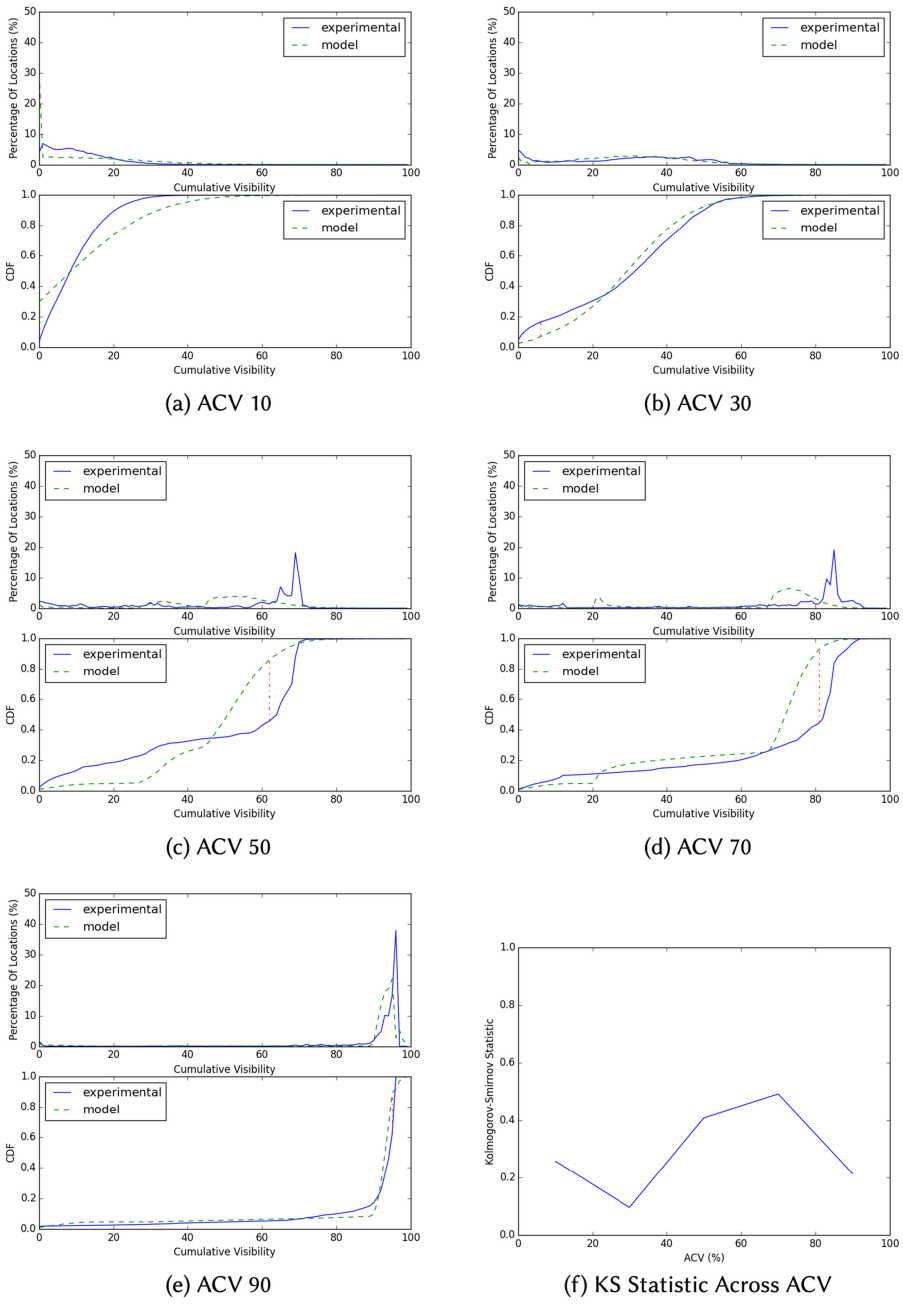

Fig. 7. Model distributions compared to experimental.

peak, is calculated based on the desired ACV. We also observe that as the ACV increases the spread around the large peak decreases, meaning a larger number of locations have similar visibility as there is less spread.

We then chose to model the fixed small peak using a Gaussian with a fixed mean and variance, i.e., the small peak is the same regardless of ACV. To adjust the contribution of each Gaussian to the

overall distribution, we use a ratio to determine how frequently we draw from each Gaussian. For example, if we set a ratio of 2:1 between the large- and small peaks, then the former will be drawn from twice as often. The net effect is that when graphed as a Cumulative Visibility distribution the large peak will be twice as big as the small peak.

The large peak's mean and variance are calculated as a function of the ACV. The mean is equal to the ACV, and the variance decreases linearly as the ACV increases to have the effect that the large ACV peak becomes narrower as the ACV increases.

Let us represent a Gaussian function using the notation $\phi(\mu, \sigma)$, where μ is the mean of the Gaussian function and σ is the standard deviation of the Gaussian function. And at each calling of the Gaussian function, a random sample is drawn from the Gaussian distribution defined by μ and σ. Let us also define a function $p()$ that generates a uniformly distributed random number between $[0, 1)$. We can then define a function $Generate\_CV(acv)$ that generates a Cumulative Visibility given an ACV as follows:

$$Generate\_CV(acv) = \begin{cases} \phi(5, 5) & \text{if } p() < \frac{1}{21} \\ \phi\left(\frac{21}{20}acv - \frac{5}{20}, 20 - \frac{2}{10}acv\right) & \text{if } p() \geq \frac{1}{21} \end{cases}, \tag{1}$$

where the constants 21/20, 5/20, 2/20, and 1/21 were determined empirically,

A Cumulative Visibility is assigned to each location by calling the $Generate\_CV$ function given an ACV. But the Cumulative Visibilities may not be feasible. For instance, if one location on the map is assigned a Cumulative Visibility of 0% and the remainder of the locations are assigned 100%, then it is obviously not feasible to achieve both. We handle this by doing a best-effort attempt by sorting the locations from highest to lowest Cumulative Visibility, picking the location with the highest Cumulative Visibility that has not yet been assigned links between other locations and attempting to create a link between this location and the remaining locations in the sorted location list in order of highest to lowest Cumulative Visibility. A link is formed if both locations are below their assigned Cumulative Visibility, and a location is removed from the list if it reaches the assigned Cumulative Visibility or has attempted to form a link with all remaining locations in the sorted list.

We go through a few additional steps to make our model accurate. For instance, the Wang Viewshed Algorithm forces all locations to have visibility of their immediate eight neighbors. It also requires all links to be symmetric, i.e., if location A is visible from location B, then location B is visible from location A. Our model also enforces these rules. Using our $Generate\_CV(acv)$ function with the rules described above, we can generate a modeled terrain map given an ACV as follows:

(1) Iterate through every location on the map and assign a Cumulative Visibility based on the function $Generate\_CV(acv)$ for the respective ACV and create a link between all immediate neighbors.
(2) Sort locations from highest to lowest Cumulative Visibility.
(3) Choose the location with the highest Cumulative Visibility and attempt to form a link with all remaining locations. Create a link if both locations are below their respective Cumulative Visibility threshold.
(4) Remove current location from sorted list and repeat step 3 until there are no more locations.

Figure 7 plots the Cumulative Visibility distribution resulting from our model against the Cumulative Visibility distribution of actual terrain maps with a given ACV. We compare the model distribution to the experimental using the Kolmogorov–Smirnov Statistic, which is the largest distance when comparing two **Cumulative Distribution Functions (CDF)**. In our case, the CDF is the Cumulative Cumulative Visibility distribution, which means a point represents the chance a Cumulative Visibility drawn from the respective Cumulative Visibility distribution is less than or equal to the Cumulative Visibility selected. The Kolmogorov–Smirnov Statistic is also plotted in

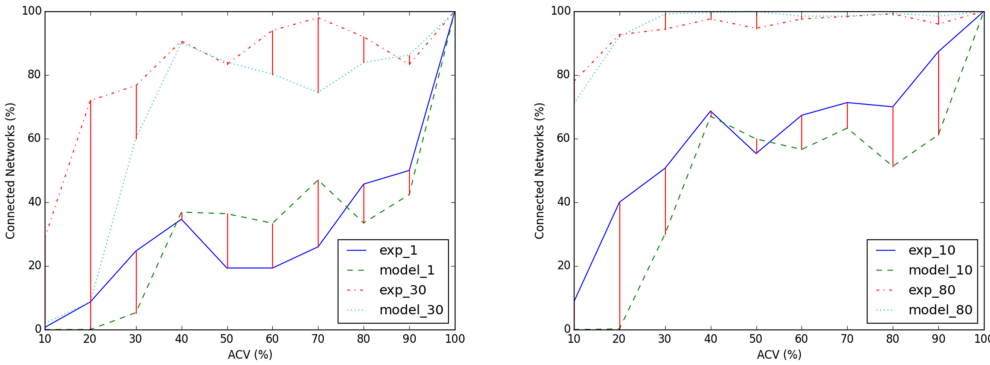

Fig. 8. Connectedness comparisons between model and experimental data.

Figure 7 as the red line that appears in the CDF graphs. We achieve less than a 0.5 Kolmogorov–Smirnov Statistic across ACV as shown in Figure 7(f).

The "experimental" curves are obtained by averaging three terrain maps with 100 by 100 elevation points, while the "model" curves plot the Cumulative Visibility distribution resulting from the model we created.

We use the proposed Cumulative Visibility distribution–based connectivity model to check network connectedness. We first generate a map based on our model and then randomly place nodes on the map following a given *density*, i.e., the number of nodes per map location. For instance, a density of 1% means there is one node for every 100 map locations. As discussed in Section 5, we set the nodes' transmission range greater than the length of the map's diagonal allowing every node the possibility of communicating with every other node if they are visible to one another based on TerrainLOS. This is a best-case scenario where only the terrain plays a role in establishing connectivity.

We then say that the resulting network is connected if every node has a path to every other node. Figure 5(a) plots the average network connectedness as a function of the ACV for different densities, where each data point is the average of 1,000 random layouts. In Figure 8, we compare network connectedness resulting from our model against results obtained experimentally, as explained in Section 5. We observe that, even though there are discrepancies between experimental and predicted network connectedness per our model, as shown by the error represented by the vertical red lines for the different data points in the graph, network connectedness resulting from our model tends to exhibit similar trends as experimental connectedness.

As part of future work, we plan to compare the performance of routing protocols using our ACV model to generate terrain against actual terrain with a given ACV.

7 PERFORMANCE OF IOT ROUTING OVER TERRAIN

In this section, we evaluate how terrain affects the performance of network routing. In particular, we compare RPL [29] and ORPL [12], two well-known sensor network and IoT routing protocols. In fact, RPL is considered the current de facto routing protocol standard for the IoT. We start by briefly describing the two protocols and highlighting the differences between them.

7.1 RPL and ORPL

RPL was designed for sensor networks where the majority of nodes are sensing data and periodically reporting to a sink node. In these types of networks, energy is critical, as the sensor nodes

are generally battery powered. There is usually a tradeoff made between energy conservation and latency.

In RPL the routing layer is formed as a tree, where the sink is the root. What follows is that every node has one parent. As long as data are being reported infrequently, as is the typical case in sensor networks, the parent nodes will not have to drop packets. In addition, energy is conserved as the only nodes that consume power when delivering a packet are on the path to the sink. We will see how ORPL uses a different method.

ORPL [12] is an opportunistic routing protocol. Unlike RPL, ORPL allows nodes to have multiple parents. In a dense network ORPL should perform better as the routing layer can take advantage of the opportunistic nature of the network. One would assume this would improve latencies but at the cost of energy consumption. But in the paper published by Duquennoy et al. [12], the authors showed that ORPL not only improved latency but also decreased energy consumption. We will discuss the experiments performed in their paper as well as how our simulations differ when we discuss our results.

Using TerrainLOS, we can compare these two routing protocols over terrain and examine when it is better to use RPL or ORPL. These types of simulations can be applied to any two protocols, but we use RPL and ORPL as a case study.

7.2 Experimental Methodology

Most evaluations of networks and their protocols designed for outdoor deployment do not account for terrain unless they are deployed in the field. The goal of our experiments are to consider the effects of terrain when testing and evaluating core network functions such as routing in network simulation/emulation platforms. To demonstrate the role of terrain-aware communication on network performance, we use our implementation of TerrainLOS in COOJA (as described in Section 3) and compare the performance of two well-known IoT routing protocols, namely RPL and ORPL, both of which are described below. As noted previously, since TerrainLOS targets outdoor IoT deployments, we implemented it in the COOJA experimental platform, as it is designed to simulate/emulate IoT networks and their protocols. COOJA also implements both RPL and ORPL. While we implemented TerrainLOS in COOJA, it can be ported to any other network simulation/emulation platform.

Our simulations were inspired by Duquennoy et al. [12] where ORPL was first proposed, and we use the results from their paper as a reference point for our experiments and results. We also ensure that all networks used in our runs are 100% connected. This is a new methodology when compared to the experiments run in our previous paper. Duquennoy et al. performed their experiments using the Indriya testbed [10], which uses 139 TelosB motes (135 motes were available during their experiments) running the Contiki operating system spread across three floors of an office building. The authors compared ORPL against RPL, CTP, ORW, and LWB using both low average degree and a high average degree scenarios, where degree is given by the number of direct links (or edges) connecting a node (or a vertex) to other distinct nodes.

The authors also compare three different routing scenarios: upwards routing, downwards routing, and any-to-any traffic. All experiments were run for one hour. The Upwards routing scenario is when nodes send traffic at a given periodic interval to a data collection site called the *sink*. The downwards routing scenario is when the sink node sends traffic at a given periodic interval to a random node. The any-to-any traffic scenario is when a random node sends to a different random node at a given periodic interval.

In our experiments, we examine the upwards routing application where all nodes send to a sink node, since it is the most typical sensor network/IoT scenario. Every node sends a 64-B packet periodically at a uniformly random time in the subsequent four minute window to a sink node.

Table 2. Experimental Setup

	Let The Tree Bloom	TerrainLOS Corner	TerrainLOS Center
Motes	135	100	
Experiment Type	Indriya Testbed	COOJA Simulation	
Application	Upwards		
Packets/sec	0.56 packets/s	0.41 packets/s	
Layout	Indriya Testbed	Uniform Random	
Duration	One Hour		
Low Average Degree (30% ACV)			
Diameter	9.8	11.3	9.0
Degree min	1.0	1.0	1.3
Degree avg	9.3	9.3	10.6
Degree max	24.0	18.0	21.7
Mid-Low Average Degree (60% ACV)			
Diameter	NA	7.7	8.7
Degree min	NA	1.3	1.3
Degree avg	NA	10.5	10.9
Degree max	NA	21.3	22.3
Mid-High Average Degree (90% ACV)			
Diameter	NA	6.3	6.3
Degree min	NA	3.7	3.3
Degree avg	NA	16.7	15.7
Degree max	NA	27.0	26.0
High Average Degree (100% ACV)			
Diameter	5.7	6.0	6.0
Degree min	3.0	5.0	4.0
Degree avg	17.1	17.4	17.4
Degree max	33	27.0	27.0

We use the same implementation of RPL and ORPL, which was provided by the authors at https://github.com/simonduq/orpl. COOJA's MAC layer uses the ContikiMAC with a wakeup interval of 500 ms, which is the amount of time a node sleeps before waking up and forwarding messages. It also determines the length of time a node must transmit to guarantee that a node will hear the transmission and forward the message if applicable. COOJA's ORPL uses a weight of 0.5 as that was found to have the best performance by Duquennoy et al. The ORPL weight is a value typically from 0 to 1 that determines whether a node forwards a message. This parameter is needed for ORPL and not RPL as ORPL is opportunistic and any node within transmission range of a given node can potentially forward the message. The weight is essentially a way to make sure that only nodes closer to the destination will forward the message.

The difference between our experiments and the ones conducted by Duquennoy et al. [12] is that we are simulating using COOJA instead of using a testbed. We also use 100 nodes instead of 135. By using 100 nodes, we bring the network load to 0.41 packets/second instead of 0.56 packets/second used in the Duquennoy et al. paper. We vary the ACV of the simulation to change the network average degree as opposed to the transmission power, which was used by Duquennoy et al. A summary of our experimental setup highlighting the difference between our experiments and the ones conducted in Duquennoy et al. is provided in Table 2. In addition, the Indriya testbed is an

indoor testbed instrumented in a multi-story building; this means it is possible to have multiple nodes at different heights at the same (x, y) coordinate. This is not possible using TerrainLOS. If two nodes are at the same (x, y) coordinate in TerrainLOS, then they also have the same height.

We use the Duquennoy et al. results as reference, but our main goal is to show how we can use TerrainLOS to evaluate protocols over varying terrain. As such, we do not expect our results to match the experiments by Duquennoy et al. as the experimental methodology and scenarios are different.

We chose to use a density of 30%, i.e., there are 30 nodes for every 100 map locations. We chose a density of 30%, as it has a similar range of average degree across ACV to compare to the ORPL results. We then vary the ACV to get a range of average degree. We only use networks that are fully connected. What we expect is that at low ACV, where the average degree is lower, RPL will have better performance in general, and at high ACV ORPL will have better performance, since the average degree is higher, which means there is a larger amount of opportunism.

Multiple simulations were run with 100 randomly placed stationary nodes for 1 hour of simulated time. Each node's transmission range was set to the diagonal of the simulation area, which may not reflect real-world deployments but was chosen to achieve a connected network as discussed in Section 6. In our results, each data point is an average over all 100 runs.

We ran two scenarios, one where the sink is placed in a corner of the simulation area, a typical sensor network layout where the sink is located in an area with infrastructure and the remaining nodes are placed in an area that is infrastructure-less, such as a forest. The other scenario has the sink in the center of the simulated area, where the sink would have the smallest distance to every node.

We analyze five different metrics: **packet delivery ratio (PDR)**, latency, energy consumption, duplicates, and transmissions. We measure PDR by counting the number of packets received divided by the number of packets sent. We measure energy consumption based on the percentage of time the radio is on for each node, which in previous work was shown to correlate directly with energy usage and is platform independent [12]. We measure duplicate traffic by recording at each hop per packet the amount of transmissions for the packet to be transferred, which includes collisions. If a future transmission reports a hop previously recorded, then this is counted as duplicate traffic. Duplicate traffic is also counted if a transmission reports a hop greater than the number of hops it takes to reach the destination. Transmissions is the total number of transmissions per simulation.

7.3 Results

We plot the five metrics as a function of the ACV in Figures 9, 10, and 11. We use a range of ACVs, namely 30%, 60%, 90%, and 100%. Each data point is averaged over five or more runs, and the error bars represent one standard deviation.

By varying the ACV, we show how terrain affects routing protocol performance. As reference, we compare our results to the original ORPL paper, when data were available (labeled "ltb_" for Let the Tree Bloom, the title of Duquennoy et al.'s paper). Note that often only one or two data points were available. The ORPL paper was not run over ACV classified terrain, so we instead correlate points on our graphs based on the average degree of the experiments, which can be seen in Table 2.

The PDR, shown in Figure 9, remains above 96% across all runs using either RPL or ORPL as the routing layer. When compared to Duquennoy et al., our results show that RPL and ORPL have very similar PDRs, except at 30% ACV in the corner sink layout, where ORPL has a drop in PDR, although there is a 4% standard deviation between runs. Duquennoy et al. show RPL has a consistently lower PDR, which differs from our results. One thought on why our results differ could have to do with random drops in the network. When we simulate, we always use a 100%

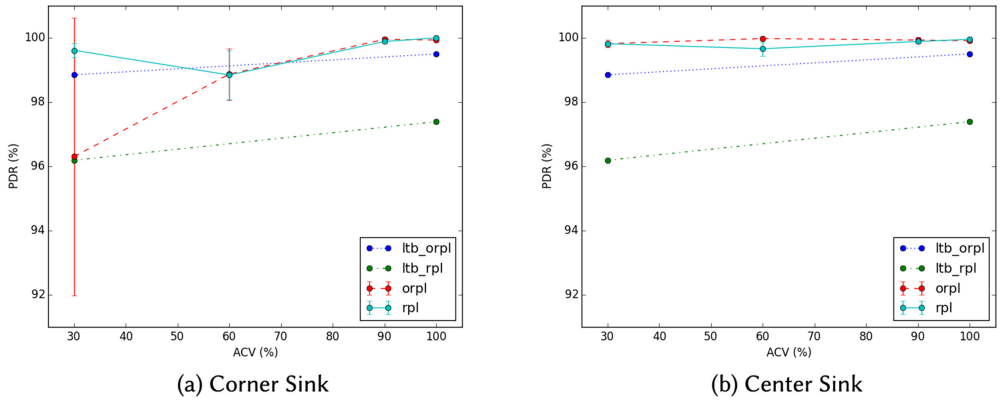

Fig. 9. Packet delivery ratio analysis.

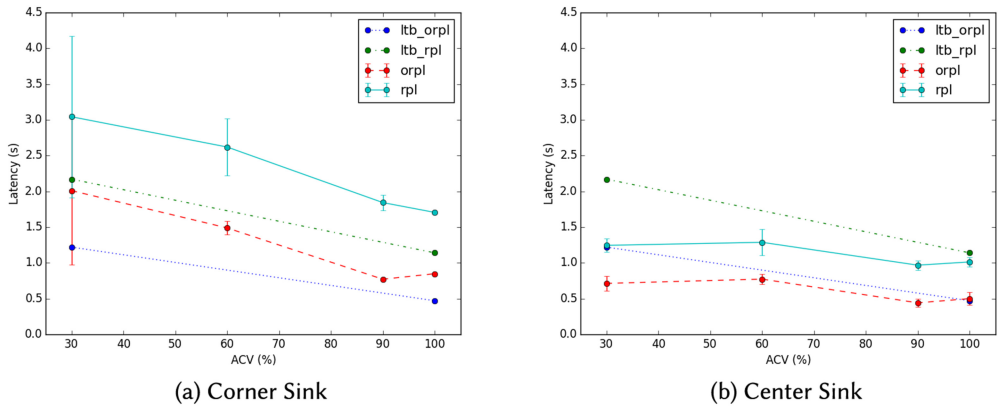

Fig. 10. Latency analysis.

transmit and receive success rate. In other words, in our experiments, the only way a packet will be dropped is because of a collision.

If we instead consider transmission losses, then it is possible our results would more closely resemble the results from the Duquennoy et al. paper. However, we should again point out that we are using Duquernoy et al.'s results only as reference and not necessarily as a comparison baseline, as our experimental platforms are considerably different. Furthermore, our goal here is to demonstrate how terrain can significantly impact network performance for outdoor deployments and thus the importance of accounting for terrain when evaluating networks and their protocols.

Figure 10 plots our latency results, which are exactly as we expect. ORPL consistently yields lower latencies as it can take advantage of opportunism. Note that as the ACV decreases, which decreases the average degree and therefore the opportunism, latency increases. The corner sink scenario has higher overall latencies as the average distance to the sink is larger as a whole. Our latency results are comparable to Duquennoy et al.

Energy consumption, plotted in Figure 11, increases as the number of transmissions increase. As the transmissions increase ORPL much more rapidly has increased energy consumption. ORPL also has a higher rate of transmissions as it is opportunistic, which can be seen as the amount of duplicates increase with ACV. These results differ than those presented by Duquennoy et al. and

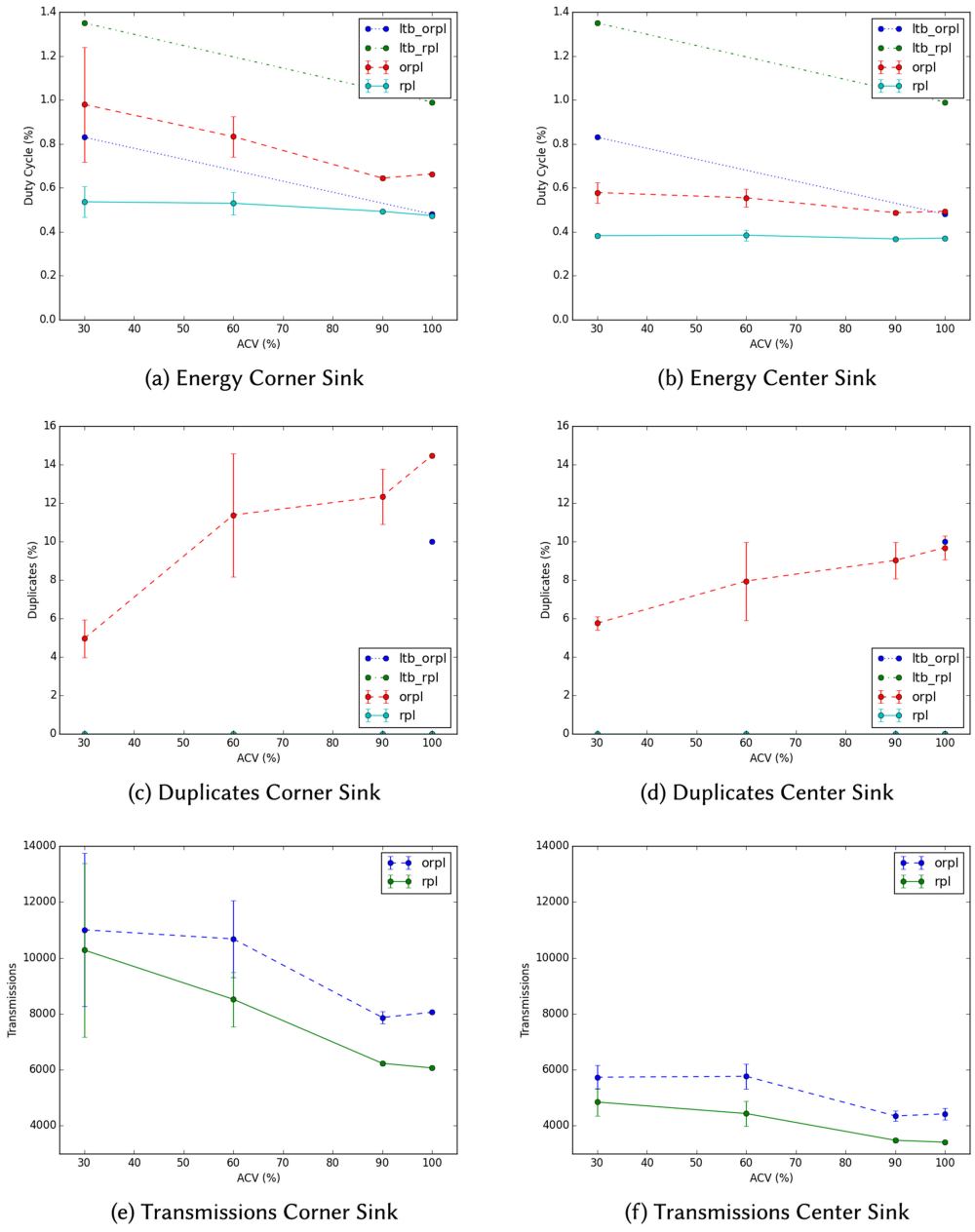

Fig. 11. Energy, duplicates, and transmission analysis.

in fact are the opposite. We find that ORPL uses more energy than RPL in all our scenarios. This could be from the reduced network load or because the simulations are not taking into account random drops. As expected, we do see that in the scenarios where the sink is in the corner, more energy is consumed, as packets on average have a longer path to follow.

All the metrics we measure do vary with ACV. This makes sense, because at lower ACVs, average degree per node is lower, which affects routing performance. We thus show that terrain should

be an important consideration when evaluating networks, their applications, and core functions intended for outdoor deployment. We argue that TerrainLOS is an excellent propagation model to use when evaluating IoT core networking functions and applications using simulations, as it takes terrain into account and does not significantly affect simulation time or memory overhead as shown in Section 3.3.

8 CONCLUSION AND FUTURE WORK

In this article, we introduced a new propagation model called TerrainLOS that uses real terrain height data to determine the connectivity of a network. We implemented our model in COOJA in order for IoT applications to simulate outdoor scenarios more accurately.

We created our own classification of terrain using ACV and proposed a novel model to generate a terrain given an ACV. Using TerrainLOS, we compared network connectivity using our terrain model against real terrain with the same ACV and found similar trends. We used TerrainLOS to evaluate two IoT routing protocols, RPL and ORPL, and compare their performance over various classes of terrain. We find that when compared to RPL, ORPL is able to achieve lower latencies at a cost of higher energy consumption across terrain.

In future work, we plan to compare TerrainLOS against existing signal strength data, e.g., Reference [19]. We will also explore how to compare TerrainLOS results with existing implementations of terrain-aware propagation models such as those listed in Section 2. Additionally, we plan to compare routing protocols using our model of terrain for different ACVs against using real terrain with the same ACVs.

REFERENCES

[1] [n.d.]. The Network Simulator—ns-2. Retrieved from https://www.isi.edu/nsnam/ns/.

[2] [n.d.]. OMNeT++ Discrete Event Simulator. Retrieved from https://omnetpp.org/.

[3] [n.d.]. OPNET Is Now art of Riverbed SteelCentral. Retrieved from https://www.riverbed.com/products/steelcentral/opnet.html.

[4] [n.d.]. Planetside Software. Retrieved from https://planetside.co.uk/.

[5] [n.d.]. Shuttle Radar Topography Mission. Retrieved from https://www2.jpl.nasa.gov/srtm/statistics.html.

[6] [n.d.]. Terraform Home Page. Retrieved from http://firedrake.org/terraform/.

[7] Alexandru Bumb, Bogdan Iancu, and Emil Cebuc. 2018. Extending cooja simulator with real weather and soil data. In *Proceedings of the 17th RoEduNet Conference: Networking in Education and Research (RoEduNet'18)*. IEEE, 1–5.

[8] K. T. Chang. 2012. *Introduction to Geographic Information Systems* (6th ed.). McGraw–Hll.

[9] A. S. Joseph Charles and P. Kalavathi. 2018. QoS measurement of RPL using cooja simulator and wireshark network analyser.

[10] Manjunath Doddavenkatappa, Mun Choon Chan, and Akkihebbal L. Ananda. 2012. Indriya: A low-cost, 3D wireless sensor network testbed. In *Testbeds and Research Infrastructure. Development of Networks and Communities*. Springer, 302–316. http://link.springer.com/chapter/10.1007/978-3-642-29273-6_23.

[11] Adam Dunkels, Björn Grönvall, and Thiemo Voigt. 2004. Contiki-a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*. IEEE, 455–462. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1367266.

[12] Simon Duquennoy, Olaf Landsiedel, and Thiemo Voigt. 2013. Let the tree bloom: Scalable opportunistic routing with ORPL. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2. http://dl.acm.org.oca.ucsc.edu/citation.cfm?id=2517369.

[13] R. Edwards and J. Durkin. 1969. Computer prediction of service areas for vhf mobile radio networks. In *Proceedings of the Institution of Electrical Engineers*, Vol. 116. IET, 1493–1500.

[14] Sonja Filiposka and Dimitar Trajanov. 2010. Terrain-aware three-dimensional radio-propagation model extension for NS-2. *Simulation* (July 2010). https://doi.org/10.1177/0037549710374607.

[15] John Heidemann, Nirupama Bulusu, Jeremy Elson, Chalermek Intanagonwiwat, Kun-chan Lan, Ya Xu, Wei Ye, Deborah Estrin, and Ramesh Govindan. 2001. Effects of detail in wireless network simulation. In *Proceedings of the SCS Multiconference on Distributed Simulation*. 3–11. http://www.isi.edu/~johnh/PAPERS/Heidemann00d.pdf.

[16] Lance Joneckis, Corinne Kramer, David Sparrow, and David Tate. 2014. Modeling terrain impact on manet connectivity. In *Proceedings of the Military Communications Conference (MILCOM'14)*.

[17] Mohan J. Kumar, P. R. Venkateswaran, and Gopalakrishna N. Kini. 2018. Software-based energy estimation for localization algorithms in wireless sensor networks using Contiki-COOJA. *Int. J. of Pure and Appl. Mathe.* 119, 12 (2018), 12655–12663.

[18] Sam Mansfield, Kerry Veenstra, and Katia Obraczka. 2016. TerrainLOS: An outdoor propagation model for realistic sensor network simulation. In *Proceedings of the IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'16)*. IEEE, 463–468. http://ieeexplore.ieee.org/abstract/document/7774622/.

[19] P. L. McQuate, J. M. Harman, and A. P. Barsis. 1968. *Tabulations of Propagation Data Over Irregular Terrain in the 230-TO 9200-MHz Frequency Range. Part 1: Gunbarrel Hill Receiver Site.* Technical Report. Institute For Telecommunication Sciences Boulder CO.

[20] Calvin Newport, David Kotz, Yougu Yuan, Robert S. Gray, Jason Liu, and Chip Elliott. 2007. Experimental evaluation of wireless simulation assumptions. *Simulation* 83, 9 (2007), 643–661. http://sim.sagepub.com/content/83/9/643.short.

[21] H. H. Nguyen, S. Krug, and J. Seitz. 2016. Simulation of 3D signal propagation based on real world terrains for Ad-Hoc network evaluation. In *Proceedings of the 9th IFIP Wireless and Mobile Networking Conference (WMNC'16)*. 131–137. https://doi.org/10.1109/WMNC.2016.7543980.

[22] nsnam. [n.d.]. ns-3.

[23] Afonso Oliveira and Teresa Vazão. 2018. Connecting NS-3 with cooja. In *Proceedings of the IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD'18)*. IEEE, 1–6.

[24] Fredrik Osterlind, Adam Dunkels, Joakim Eriksson, Niclas Finne, and Thiemo Voigt. 2006. Cross-level sensor network simulation with cooja. In *Proceedings 2006 31st IEEE Conference on Local Computer Networks*. IEEE, 641–648. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4116633.

[25] Krzysztof Pawlikowski, H.-D. J. Jeong, and J.-S. R. Lee. 2002. On credibility of simulation studies of telecommunication networks. *Commun. Mag. IEEE* 40, 1 (2002), 132–139. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=978060.

[26] Erina Rodrigues and Deven Shah. 2018. Software defined network for internet of things.

[27] Ernesto Rodriguez, C. S. Morris, J. E. Belz, E. C. Chapin, J. M. Martin, W. Daffer, and Scott Hensley. 2005. An assessment of the SRTM topographic products.

[28] Sonja Stojanova, Leonid Djinevski, Igor Mishkovski, Sonja Filiposka, and Dimitar Trajanov. 2013. Micro-benchmarking NS-2 and NS-3 network simulators using terrain aware radio propagation extension. In *Proceedings of the 3rd International Conference on Internet Society Technology and Management (ICIST'13)*. 81–84.

[29] Tsvetko Tsvetkov. 2011. RPL: IPv6 routing protocol for low power and lossy networks. *Sens. Nodes-Operat. Netw. Appl.* 59 (2011), 2. https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2011-07-1.pdf#page=67.

[30] Kerry Veenstra and Katia Obraczka. 2015. Guiding sensor-node deployment over 2.5 D terrain. In *Proceedings of the IEEE International Conference on Communications (ICC'15)*. IEEE, 6719–6725. http://ieeexplore.ieee.org.oca.ucsc.edu/xpls/abs_all.jsp?arnumber=7249396.

[31] Jianjun Wang, Gary J. Robinson, and Kevin White. 2000. Generating viewsheds without using sightlines. *Photogram. Eng. Remote Sens.* 66, 1 (2000), 87–90. http://info.asprs.org/publications/pers/2000journal/january/2000_jan_87-90.pdf.

[32] Mark D. Yarvis, Steven W. Conner, Lakshman Krishnamurthy, Jasmeet Chhabra, Brent Elliott, and Alan Mainwaring. 2002. Real-world experiences with an interactive ad hoc sensor network. In *Proceedings of the International Conference on Parallel Processing Workshops*. IEEE, 143–151. http://ieeexplore.ieee.org.oca.ucsc.edu/xpls/abs_all.jsp?arnumber=1039724.

Received April 2019; revised July 2021; accepted December 2021