



Article

ViTool-BC: Visualization Tool Based on Cooja Simulator for WSN

Daladier Jabba *  and Pedro Acevedo 

System and Computer Engineering Department, Universidad del Norte, Barranquilla 080001, Colombia; pdacevedo@uninorte.edu.co

* Correspondence: djabba@uninorte.edu.co

Abstract: Evaluation and monitoring of wireless sensor networks (WSN) and the parameters defining their operations and design, such as energy consumption, latency, and stability, is a complex task due to interaction with real devices. For greater control of these variables, the use of simulators arises as an alternative. Cooja is a WSN simulator/emulator which handles the devices' controllers and multiple communication protocol implementations, such as RPL (RPL is one of the most used protocol in IoT). However, Cooja does not consider either the implementation of an energy model (it has infinite energy consumption) nor the visual behavior of the topology construction, although these aspects are crucial for effective network analysis and decision taking. This paper presents the design and the implementation of ViTool-BC, a software built on top of Cooja, which allows the creation of different energy estimation models and also to visualize in real time the behavior of WSN topology construction. In addition, ViTool-BC offers a heat map of energy consumption traces. Therefore, this tool helps researchers to monitor in real time the topology construction, node disconnection, and battery depletion, aspects to be considered in the analysis of the available routing protocols in Cooja.

Keywords: wireless sensor networks; energy consumption; Cooja simulator



Citation: Jabba, D.; Acevedo, P.

ViTool-BC: Visualization Tool Based on Cooja Simulator for WSN. *Appl. Sci.* **2021**, *11*, 7665. <https://doi.org/10.3390/app11167665>

Academic Editor: Francesco Dell'Olio

Received: 30 July 2021

Accepted: 18 August 2021

Published: 20 August 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Wireless sensor networks (WSN) are a type of IoT technology that is applied in multiple areas [1,2]. This field describes the communication of sensors distributed in areas of interest for collecting, monitoring, analyzing, and predicting environmental or external phenomena that affect the mentioned area. The applications of sensor networks include air quality aspects [3], disaster prediction [4], monitoring agriculture areas [5,6], and smart cities [7,8], among others. These sensors have many restrictions, from low energy and processing capacity to unstable connections between devices. Consequently, how the information is collected and retransmitted, considering the limitations, will determine the efficiency of the designed topology. For handling these restrictions and controlling the connections of the nodes of a wireless sensor network, several routing protocols are defined [9]. These protocols include rules and steps that govern the communication between the nodes, allowing them to optimize the use of these devices' resources.

The multiple applications of the WSN require the delimitation of communication protocols, topologies of the network, and physical specifications that adapt to the scenarios of interest. Therefore, it is necessary to understand the theoretical definitions of these protocols and topologies, which allow us to correct the problems and specific phenomena in these scenarios. Given the difficulty of assembling and controlling a testbed of physical sensors [10], simulators emerge as an alternative for validating operations of a WSN with their parameters. The use of simulators on IoT into the deployed WSN helps with the measurement and estimation of the metrics, allowing the evaluation of the performance and quality of service (QoS) of a designed network or its topology control [11,12]. Some of these used metrics in WSN include energy consumption, latency, bandwidth, stability, and network lifetime, among others [13]. Also on Cloud computing, these metrics extend

their scope with the inclusion of the monitoring strategy and security aspects [14,15]. The motivation of this paper is to offer researchers the alternative of having a tool that allows deep analysis of the network behavior, such as energy consumption, packet loss, and parent's changes, among others. In this paper, the ViTool-BC (visualization tool-based Cooja simulator) is described. This tool is oriented to the set of the parameters and metrics arising from a wireless sensor network simulation generated by the Cooja simulator [16]. The main contributions of ViTool-BC are obtaining information related to the consumption and monitoring of energy, a heat map of energy traces, a battery depletion model, the latency of the sending packets, the parent's changes, and the change in the topology in real time in the simulation. The rest of the article is organized as follows: in Section 2, the related works are presented, Section 3 describes the energy model, in Section 4, the design and the implementation of the proposed tool ViTool-BC are described, and finally the conclusions and references are shown.

2. Related Works

Multiple tools and simulators have been defined and developed to emulate the behavior and remotely visualize connections of a network topology [17,18]. These simulators require extensive knowledge and hours of handling them for the deployment of test scenarios. OPNET Simulator [19] is defined as a tool for WSN simulation, and the behavior of any network, including sensor networks, is programmed under C/C++ and it is divided by domains for managing the functionalities, such as Network, Process, and Node structures. This simulator does not allow modification of the protocols already implemented. NS simulators are a series of different tools proposed by the community (open-source) for network control (NS-2 and NS-3). They are event-based discrete simulators, written mainly in the C++ language [20], and NS has multiple network protocols implemented in different layers (TCP, DSP, FTP, etc.). Due to the open-source philosophy, the NS series of simulators contain multiple configurations and extensions proposed to allow more or better simulations in several specific scenarios [21–23].

OMNET++ is an open-source discrete event simulator based on components, written on top of frameworks and C++ libraries [24]. Its modular composition allows the intuitive interaction of its functionalities, from the GUI interface to the delimitation of network protocols [25]. In the same way, this simulator has been extended [26,27]. TOSSIM is defined in [28], a simulator written in NesC (extension of C) and oriented to the functions of IoT and WSN. This tool is used to simulate devices based on the TinyOS system, allowing the simulation by delimiting the actual device drivers themselves. TOSSIM does not capture network parameters such as CPU and power consumption, and some tools or plugins have been proposed to obtain them [29]. The case of mTOSSIM [30], software that extends the TOSSIM simulator, allows the monitoring of a simulation's energy and battery consumption and also offers an external graphical environment that executes TOSSIM simultaneously, displaying the reading of these variables.

On the other hand, the Gbest-WSN simulator is described in [31]. It is a simulation tool oriented towards education since it allows the exploration of the environment, addition of protocols, and comparison of metrics without modifying or learning about the tool's code, unlike others. Gbest-WSN is written in Matlab and allows simulation of static, movable, homogeneous, and heterogeneous WSN protocols. This tool is not available for use. Another tool developed for educational purposes is presented in [32], where the Wireless GINI tool is defined. This tool is an extension of the network emulator GINI toolkit but with the addition of the ability to analyze WSN topologies. The wireless GINI tool allows to setup process-oriented networks and testbed environments with wireless support in an interactive manner, showing examples of multiple topologies and Internet architectures. Another simulator named CloudSim is oriented towards an IoT scope and is presented in [33]. CloudSim simulator is a tool for modeling the cloud computing and infrastructure services that focus on deploying deployment architecture. It offers an option to analyze the management resource usage and their policies and the data center energy

consumption. CloudSim also has other extensions allowing the user to execute more specific simulations, such as CloudSimSDN-NFV [34] and iFogSim [35]. CloudSimSDN-NFV is a tool oriented towards Software Defined Networks and their capabilities. The simulator iFogSim allows modeling of the fog computing scenarios in order to analyze and modify the resource management techniques in latency, network congestion, energy consumption, and cost.

Cooja [16] is an emulation/simulation tool for WSN based on the ContikiOS operating system [36]. This tool is a simulator written in Java, which allows the simulation of multiple IoT sensors, communication, and routing protocols. Some research has focused on extending the functionality of the Cooja simulator [37,38]. Taking advantage of the potential of this, they extract information and metadata to achieve other analysis objectives as described in [39], and research which plugins and APIs available to Cooja are used to compile relevant information about weather and surface data. A graphical view is available to show the results of the simulations. In [40], a tool is defined that integrates Cooja with an application based on TinyOS to deploy cyber-physical devices in management and industrial monitoring. Similarly in [37], the implementation of a graphical environment is described, allowing Cooja to run the simulation of a smart house and operate the protocol on closed environments (building, house, etc.).

Additionally, software that model WSN behavior or capabilities are used to extend simulation possibilities. This is the case of [41], where the authors present Viptos, which is a visualization tool for TinyOS nodes related to the TOMSSIM simulator. This tool employs the options presented in VisualSense [42], that at same time is based on the software Ptolemy II [43], where it is offered different kinds of classes and methods for sensor control and communication emulations. Viptos, through the nesC language and PtinyOS [44] framework, integrates VisualSense and TOMSSIM for the running of TinyOS programs (also in hardware devices), allowing the packet-level simulation, the models' additions for building architecture, and so on.

In [45], the NetTopo tool is presented, a Java-based visualization/emulator tool oriented for network algorithm implementations and evaluations. It provides a GUI to show routing algorithms steps, path discovery, and network organization, among other processes. NetTopo is an independent software, meaning that is not based on other simulations' tools, but its implementation contains all the logic for simulating WSN. With the same approach of independency, Netviewer [46] is described. This tool supports the topology and management of communication protocols. Additionally, in different types of environment scenarios this tool presents packets analysis such as volcanic activities and forest analysis, among others. Netviewer is constituted with a replay module for historical review with the visualization of topology and log files of the simulation.

In [47], authors define WiseObserver as WSN visualization tool written in C# and oriented primarily into the results and interpretation of the data generated through a simulation. This tool supports the functionalities of charts by node, interpolation maps, evolution videos, and reports. Among the diversity of state-of-the-art tools and simulators available, ViTool-BC, a Cooja-based tool, is presented to display the information, data, and results of a WSN simulation for the analysis of routing protocols. In addition, this tool allows the extraction and monitoring of Cooja simulations, in terms of the metrics and parameters of interest based on energy estimation models, topology construction visualized step by step, heat maps of energy consumption, and latency calculation, among others. A comparison of different solutions proposed in the literature are described in Table 1.

Table 1. WSN Visualization tools comparison.

Tool	Programming Language	Support Platforms	Advantages	Disadvantage
Viptos [41]	C y nesC	Tiny OS stack and Ptolemy II environment	-Additions of models through Ptolemy II. -It is an editing environment over nesC files. -Packet level simulation.	-TinyOS and TOMSSIM dependency. -Multiple compilation files and base programs for execution -Routing protocols and its visualization is not supported.
NetTopo [45]	Java	Not specified, but considered the used of external Wrappers	-Control of multiple network parameters such as energy consumption, bandwidth Management, etc. -Extensibility and API support for modified the behavior of a WSN. -Support real device connection and management.	-Algorithm-oriented, not consider the mote operation itself. -Requires externals addons for support others mote platforms
NetViewer [46]	Java	No specified	-Packet analysis. -Topology construction draw. -Replay module for historical actions of the simulation.	-It is an independent tool that has not specified the real mote controllers that can be executed. -The energy consumption is not considered.
mTOMSSIM [30]	C	Tiny OS stack	-Considered the Battery level for more real simulations. -Support indoor and outdoor environments.	-Requires the TOMSSIM simulator execution.
WiseObserver [47]	C#	No specified	-Parameters charts by node. -Interpolation maps of the topology. -Evolution videos and report from simulation.	-It is an independent tool that has not specified the real mote controllers that can be executed.
ViTool-BC	Python	Contiki OS stack	-Energy trace and modelling. -Topology construction -Network parameters management by node. -Heat map of energy consumption	-Required files generated previously on Cooja -Not allow modified topology environment into the GUI.

3. Energy Estimations and Metrics

For the estimation and monitoring of the network parameters involved in a simulation scenario of a WSN, it is required to have techniques that allow these values to be captured. One of these techniques is constituted by energy consumption considerations, typical of the simulated physical devices to estimate the specific consumption per node. In this paper, the energy model described in Equation (1) is considered as follows:

$$\text{Energy (mJ)} = (\text{Transmit} + \text{Listen} + \text{CPU} + \text{LPM}) \text{ mA} * (\text{Voltage}) \text{ V}, \quad (1)$$

The model described in (1) and based on [48] describes an individual node's consumption after a certain period. The time elapses between the turning on of the mote and the report of its actual state. Each variable in the model has the following purpose: (1) Transmit, which describes the consumption by the transmission action of the node, (2) Listen, which refers to the consumption in the reception state, (3) CPU, which represents the processing cost of logical actions, (4) LPM, that describes the state transition from hibernation to powering on by the nodes, and (5) Voltage, which is the required current for the trigger of

a note These metrics are defined by the time per unit cost of the task and are described in the corresponding datasheet of the selected mode for the simulations. It is expected that by evaluating the network parameters of the simulation, it is possible to obtain or qualify the staging of network topologies and their delimited characteristics, such as the protocol routing, the design, or node locations (random or in uniform formation), and the number of required nodes.

4. Discussion

In this section, design and the implementation of ViTool-BC over the Cooja simulator is presented. In Section 4.1, the ViToolBC architecture is described, including its main components. Section 4.2 defines the implementation and main characteristics of the tool.

4.1. Architecture

The development of a multi-platform desktop application written in the programming language Python and supported by the QT graphical user interface library [49] is defined. This tool is mainly focused on the Linux operating system, due to the direct capability of Contiki OS in one of these distributions (Instant Contiki). Likewise, the results extracted from Cooja from either the Cooja compiled files or from a virtual machine can be interpreted by the tool in any OS.

It is based on the design pattern MVC, which consists of three types of objects in which the Model is the schema of the application, the View is the screen of presentation, and the Controller defines the way the user input is processed. As it can be seen, it is a highly modular and decoupled [50], and could also be described as a design pattern, such as is presented in [51,52]. For management and control of the simulation scenarios, components that describe the core of the system are presented in Figure 1. The system architecture is divided into the modular components and classes that define the system's functionalities.

4.1.1. Simulation Component

This represents the main component for the control of a simulation instance, and is the connection between the final user and the logic process of the executed tool. Among the options available, this component includes: the definition of new simulation scenarios, reproduction of log files, and nodes displayed graphically on the screen canvas. The three main subcomponents are explained as follows:

- **Mote:** It defines the class in which the methods and variables related to a node or device are available within Cooja simulation. This component includes functions and methods such as obtaining the IP address or Rime address, ID of the node, listing and filter of Parent set, energy trace in real time, and remaining battery, among others. These types of procedures allow the tool to show the detailed trace of the current state of each node.
- **Topology control:** The focus of this subcomponent is the generation of random topologies. The resulting random topologies are downloaded into an XML file with a csc extension, which permits to be evaluated in the Cooja tool. Some of the simulation's parameters included in the XML file are the values of node position, TX, and RX. The functionality of this component is restricted to the window commands.
- **Timer:** Subcomponent in which the logic and handling of the simulation time is defined, allowing control actions (play, pause, and reset) of the simulation/replay time value of the scenario. The threads and signals were used to manage and interact with the main window or GUI Thread, due to these types of complements which let the parallel executions of both tasks.

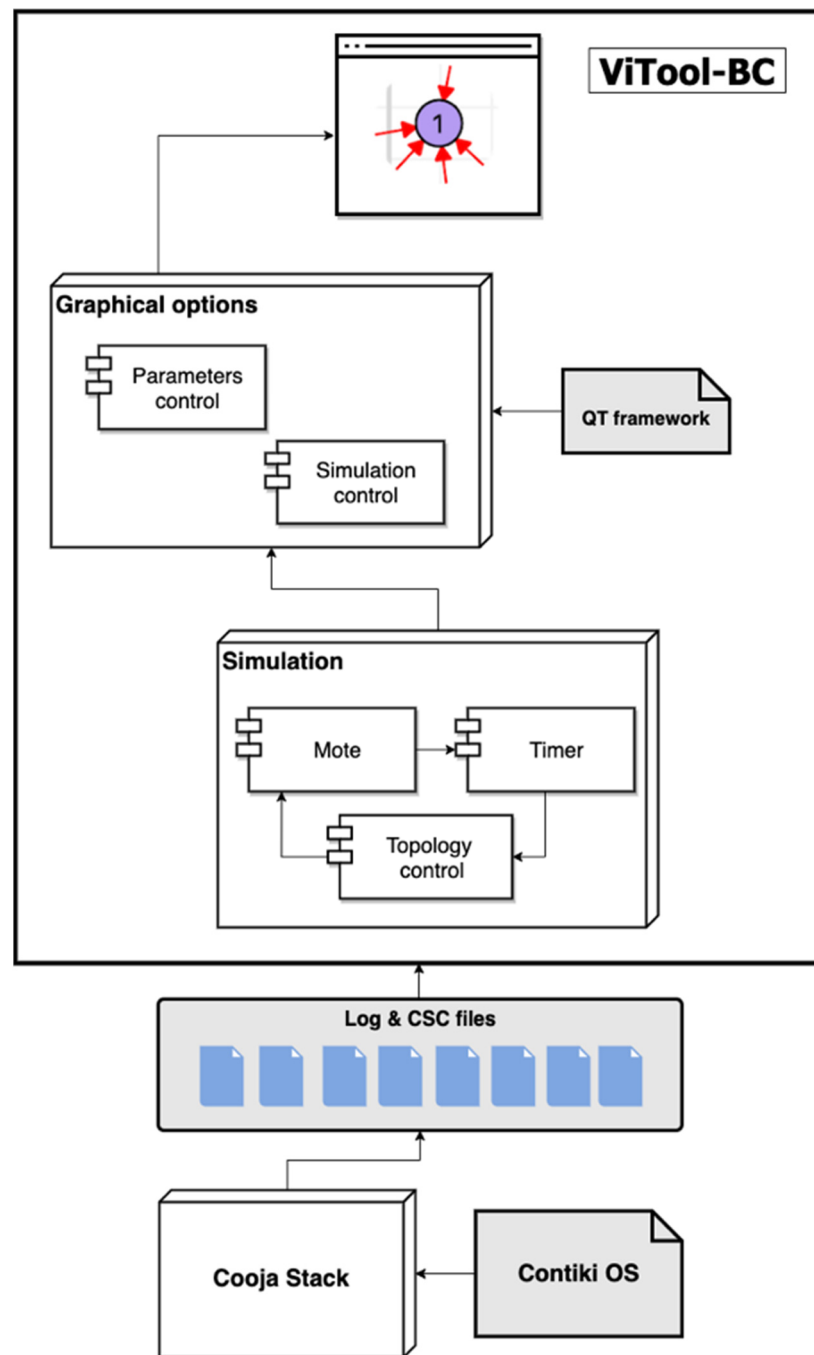


Figure 1. ViTool-BC System Architecture.

4.1.2. Graphical Options

For the interactions of the user with ViTool-BC, the platform offers different actions that interact with the simulation and its parameters and metrics. This component describes the views and their options for simulation control and user interactions. The subcomponents that define its functionalities are:

- **QT framework:** It is defined as library-oriented towards the software development that will use graphic interfaces (GUI). This library was initially written in C++, but it is available in other programming languages. The library offers routines and methods to facilitate the access and management of the user inputs and visual aspects of the developed applications.

- **Parameters controls:** There are different options to interact with ViTool-BC, and its parameters are able to manage the visual parameters defined for the simulations. This interaction delimits options such as to modify the topology scale in the canvas, to expand the size of the nodes presented in the simulation, to focus the visualization of the energy consumption in a specific mote, and to activate or deactivate visualization functionalities in the simulation process.
- **Simulation Control:** In the simulation control, components are defined functions to obtain results coming from the simulation to be reproduced. Some of these functions are: the process of managing of the Cooja files (log and csc files), options that will be applied in the simulation, a log in real time related with the executed actions in the reproduction of the simulation, and a graphical display of network topology changing in real time.

4.1.3. Cooja Stack

In the Cooja Stack, there are defined characteristics, options, outputs, and different possibilities offered by the Cooja simulator framework, including the support of different kind of motes and protocols implemented into the ContikiOS. The direct dependency of ViTool-BC with the Cooja simulator comes from data generated in the log and CSC files

4.2. Features and Main Components

ViTool-BC is defined as a visualization tool for the data generated by the Cooja simulator, in which multiple network parameters and other monitoring options are included because they are not so evident within the Cooja environment. The main view of the proposed tool is presented in Figure 2.

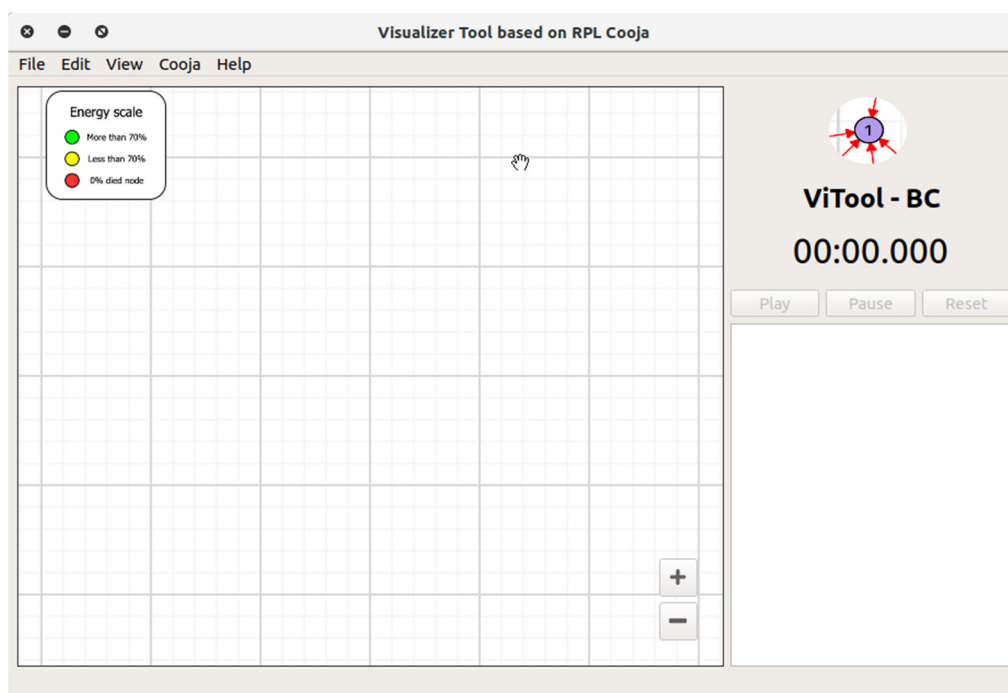


Figure 2. ViTool-BC main view.

This tool is described by modular features and functionalities, which allows access and management of the simulations generated by Cooja. From the tool, it is possible to reproduce or re-explore these simulations step-by-step in any moment, using the log files generated by Cooja and the base .csc file, in which the arrangement of the nodes on screens and the values associated with the network, such as TX and RX, are delimited.

The sequence of the reproduction of a simulation into ViTool-BC is described in Figure 3, including its main functionalities.

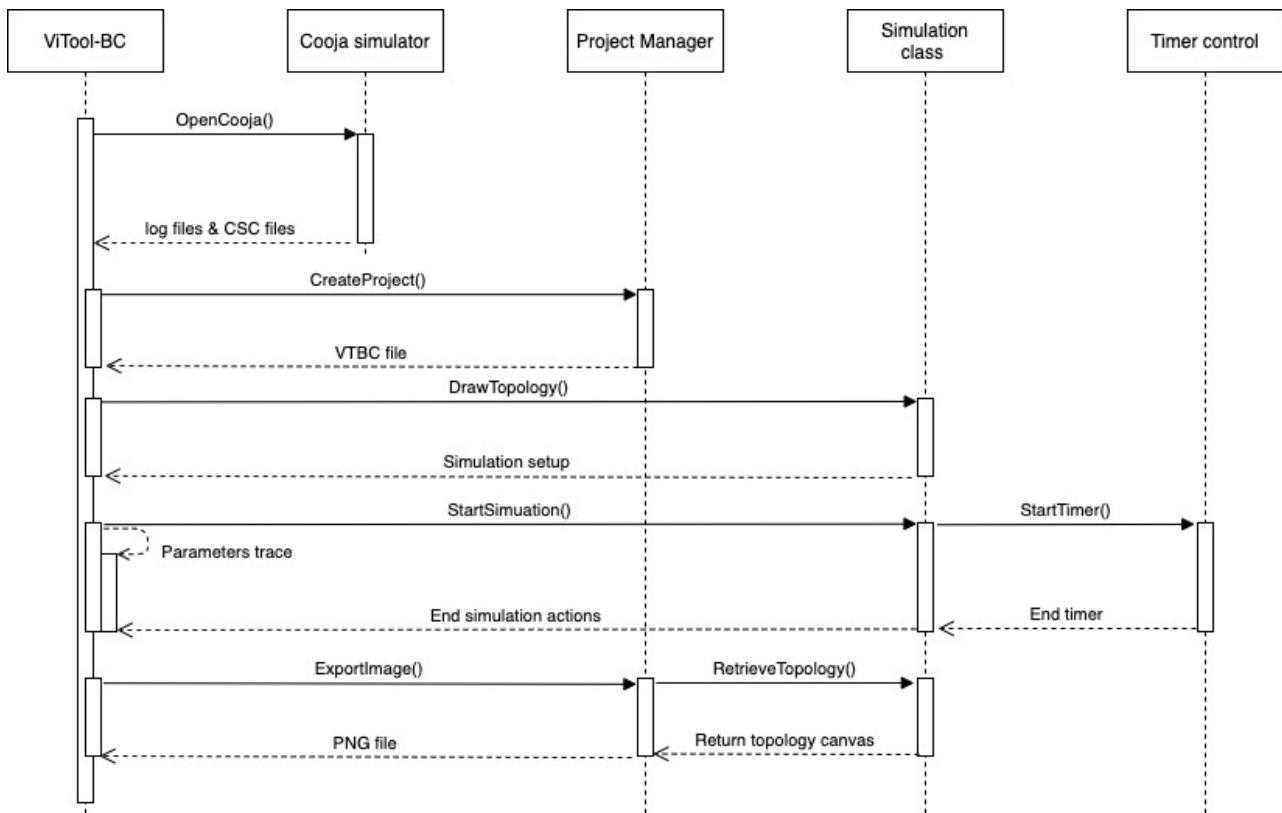


Figure 3. Sequence diagram of a simulation reproduction in ViTool-BC.

4.2.1. Project Manager

In this option, the handling of simulations is defined through projects and XML files in a chosen format (.vtbc), in which the environment variables are reflected. This vtbc file allows to reproduce the simulation (URLs in the system of the files of interest such as log and .csc) and image control parameters such as battery charge, size of the nodes on the screen, and scale in which they will be displayed into the XML file, as shown in Figure 4.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2   <simconfig>
3     <simsfiles>
4       <log>URL OF THE LOG FILE</log>
5       <csc_file>URL OF THE CSC FILE</
6     </simsfiles>
7   <parameters>
8     <energylevel>ENERGY IN mJ</energylevel>
9     <delay>DELAY FOR THE REPRODUCTION</delay>
10    <mote_size>SIZE OF THE NODE IN THE SCREEN<
11  </mote_size>
12    <node_scale>ZOOM</node_scale>
13  </parameters>
14 </simconfig>

```

Figure 4. VTBC format for project definition.

4.2.2. Timer Control

The tool allows the reproduction of the Cooja simulation; for this reason it is necessary to manage the simulation time. The buttons for time control are “Play”, which is defined as to start a simulation, “Pause”, to stop the log reading time, and “Reset”, to restart the exploration environment. It is pertinent to clarify that the timer runs and compares the actual time in action with the indicated time into the log file generated previously on Cooja. This is in order to consider the next action that the screen should display.

4.2.3. Project Manager

One of the main objectives of ViTool-BC is to offer easy access and control of the network parameters that are not evident in a Cooja simulation, such as energy trace. A color scale shows this tracking, and the color change criterion is delimited (battery green means energy stored over 70%, yellow color represents energy between 70% and 0%, and red node means without energy). In real time, the nodes change their state based on the remaining battery indicated through the tool into a simulation reproduction. These visualizations throw color possibilities into the creation of the heat map of energy of the current topology (as shown in Figure 5). This analysis illustrates the behavior of the distribution and use of the energy in each node that is directed related with the operations or tasks that a node should be doing, such as sensing, sending, forwarding, reception of packets, and messages for maintaining the network connections.

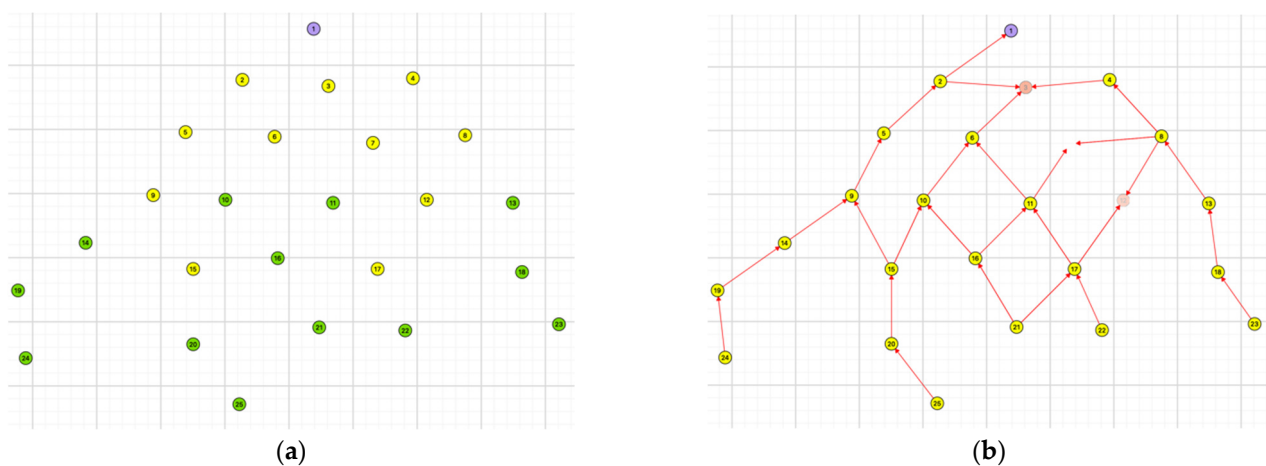


Figure 5. Heat map of energy consumption: (a) image of the scenario without visual links, where nodes near the sink are losing energy; (b) energy consumption impact, the loss of red nodes. The visualization of the topology can be exported from ViTool-BC to PNG format at any moment of the actual simulation reproduction.

4.2.4. Tree View

This view displays the node connections or tree on the screen during the simulation time (delimited by minutes), allowing researchers to identify how the nodes are connected and to estimate the network stability. One of the main challenges is to reconstruct the tree of connections into a graphical form. Some approaches were considered for drawing a graph properly, such as the BFS trip and level of each of the connected nodes.

5. Case of Use: Experiment over RPL Protocol

For interaction with the tool, the initial graphical interface is explored. Some options are disabled, such as the time control and simulation management, because a project on the canvas is not yet displayed. Before the ViTool-BC tool can be used, it is necessary to generate simulation files. This is the reason there is an option named “Cooja” in the menu of ViTool-BC with the choice “Open tool” to open the Cooja simulator. Then, from ViTool-BC, one can proceed to open the Cooja environment to perform the simulation, if

Cooja is installed. On the other hand, an error will appear, mentioning that Cooja should be installed.

In Cooja, it is required to configure the process of simulation, including the parameters to be analyzed. For this process, libraries that describe the Cooja platform and its mote are used [53]. Additionally, network protocols are considered for the simulation execution and configuration files offered by ContikiOS. Similarly, for the later reproduction of the simulation in ViTool-BC, it is necessary to consider some modifications in the source code to add in the log parameters that will be visualized in the GUI of the application. Some of them are to authorize the execution of the Powertrace program to print the energy consumption state of each node/mote and to locate code lines in the existing procedure of sending and receiving data packets. One of the examples provided in Cooja, including the collect-tree-sparse-lossy.csc, the test scenario, and its configuration, was executed. The configuration of its simulation is presented in Table 2, including information about the initial state of the execution. These considerations allow defining the test scenario in which the routing protocol is validated.

Table 2. Configuration of the tested simulation.

Parameters	Value
Operative System	Contiki 2.7
Type of node	TMote sky
Routing protocol	RPL
MAC/Adaptation layer	ContikiMAC/6LowPAN
Simulation time	30 min
Battery level considered	3500 mJ
Transmission radio	70 m
Time for periodic sent of data	30 seg
RPL parameter	MinHopRankIncrease = 256

Then, Cooja will execute the simulation with its existing options (as seen in Figure 6); the objective is validated and tests the RPL [54,55] routing protocol into a WSN topology simulation. The execution generates a follow-up in the “Mote output” window, in which a report is displayed in real time of the network actions. A large part of them must be edited and arranged by the researcher/developer.

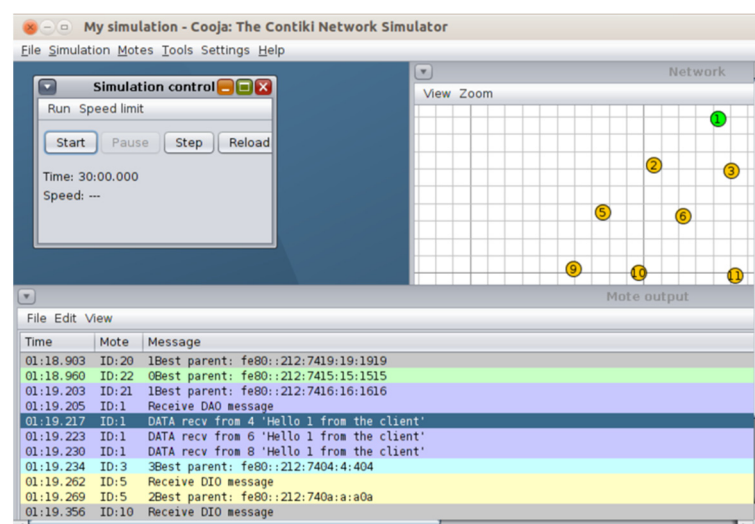


Figure 6. Cooja executes the simulation of sensors nodes.

After executing the simulation in the required time, one must proceed in the same window of the “Mote output” to export the simulation log file in a text file. Once the

simulation is finished, one must continue to return to the ViTool-BC interface in order to create a new project (Figure 7).

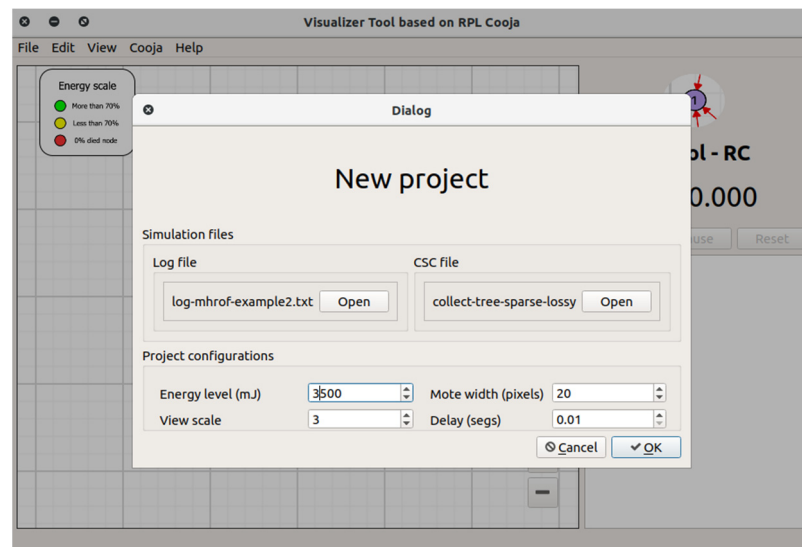


Figure 7. ViTool-BC: The new project window.

In this window, URL values of the log and .csc files (generated in Cooja), energy of the nodes (this can depend on the energy disposed of Cooja), a width of the nodes, scale of the topology, and expected delay for the time the simulation are entered. After creating a project, the simulation network design is displayed on the selected ViTool-BC environment. From the different components of the tool, the simulation can be followed-up. Then, the simulation results can be reproduced using the time control buttons, and it will start the representation of the topology actions in the graphic canvas. To conduct this process, it is necessary to add the log file with the exported data generated from the Cooja simulation, as seen in Figure 8.

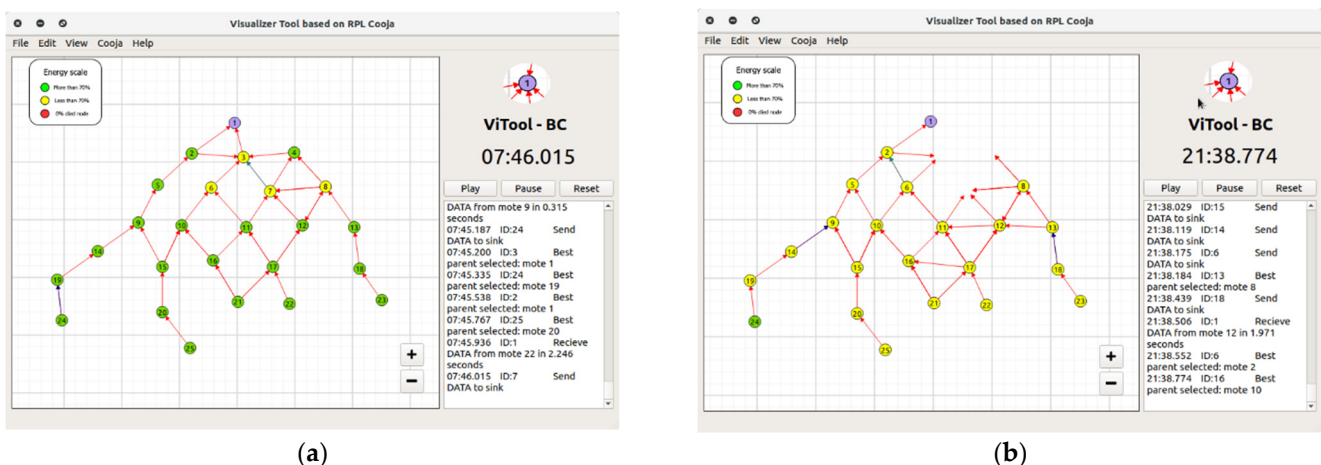


Figure 8. ViTool-BC simulation: (a) simulation reproduction started and (b) nodes without energy, removed from the topology.

The actions developed in ViTool-BC are: energy monitoring through color ranges, a direct connection between devices indicated with red lines, sensing information shown in colored lines (no red color used), dynamic displays of the topology construction. All of these are located in the graphic options, as shown in Figure 8a. It is important to take into account that an energy consumption model was built in ViTool-BC to detect the energy of

each node, due to this fundamental action not being implemented in Cooja. Another benefit of having this implementation is that researchers can modify or introduce other energy models. The log describes the remaining percentage of each node's battery and latency calculation when sending it to the sink. Thus, each of the nodes presents its information, as displayed in Figure 9.

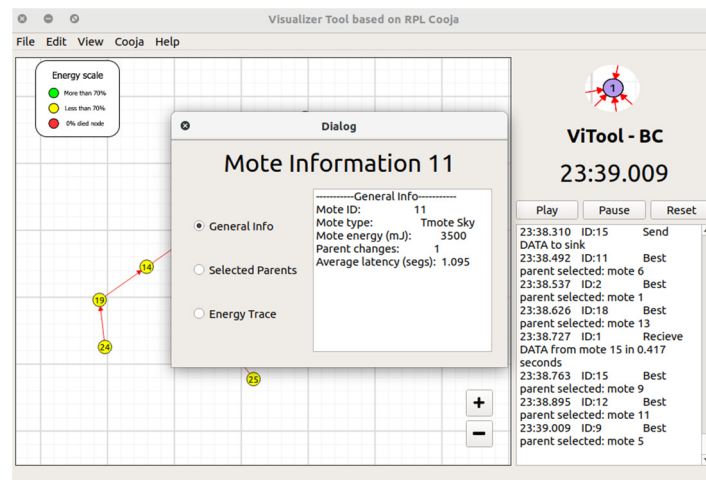


Figure 9. ViTool-BC features: mote information.

In the created simulator, nodes that lose their energy are eliminated from the graphical view (Figure 8b). Another tool's functionality is the delimitation of the topologies formed through the simulation execution that shows nodes connected in a specific time. It is described as "Tree View" (Figure 10), and the nodes are displayed on screens in a tree topology depending on their link communications.

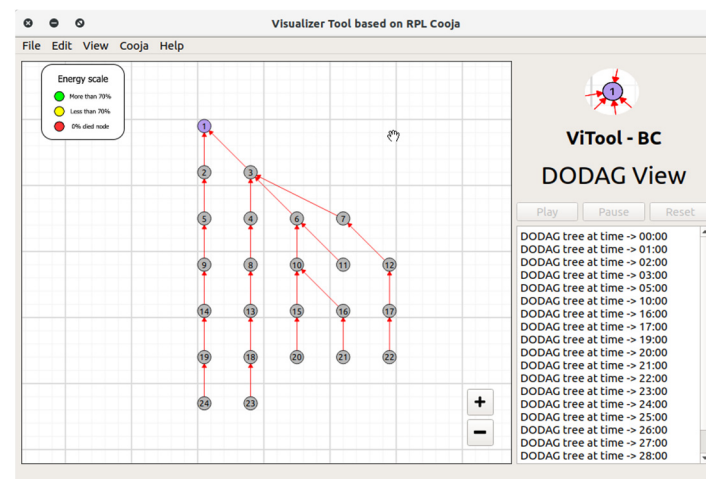


Figure 10. ViTool-BC connection tree view.

Finally, after the reproduction of the simulation with ViTool-BC, the parameters and metrics analysis allow to conclude that:

- The considered energy was not enough. More than 50% of the nodes are in the yellow state (less than 70% of battery remaining) after the half of the simulation time.
- The network stability is maintained; the parent change and communication path changes of all nodes in the network are less than three, on average. These parameters are directly influenced by the disposition of the nodes in the analyzed area.

- The latency between the nodes is, on average, less than 2 s. The node in the down level takes more time to successfully communicate with the root because multiple hops are implied.
- At the time 19:01.252, node 3 lost all of its energy and was eliminated for the topology. This information allows the estimation of the network lifetime and could be analyzed and compared between protocols and variations of them.

6. Conclusions

In this paper, the design and implementation of the ViTool-BC tool are presented as an alternative to visualizing and configuring aspects on top of the Cooja simulator, aspects which are not implemented in the Cooja tool. ViTool-BC is a graphical desktop application aimed at researchers and developers to enable them to create and implement energy estimation models, visualize WSN topology construction in real time, and create and implement heat maps of energy traces, battery level consideration, and node loss. All these options are oriented to cover the scope of the different possible analyses over a WSN device arrangement. Using ViTool-BC, it is possible to perform a more effective network analysis than is allowed from Cooja. In addition, ViTool-BC includes network parameters and metric evidence into the same GUI of the tool, information that is not found in the Cooja GUI. The authors are currently working on the inclusion of other associated network parameters such as the bandwidth, traffic management, and control of security vulnerabilities. In the same way, in the future, work is intended to detach direct dependence using the Cooja tool.

Author Contributions: P.A.: Conceptualization, software, validation, formal analysis; investigation, writing—original draft preparation; D.J.: methodology, resources, writing—review and editing, visualization, supervision, project administration. Both authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Modieginyane, K.M.; Letswamotse, B.B.; Malekian, R.; Abu-Mahfouz, A.M. Software defined wireless sensor networks application opportunities for efficient network management: A survey. *Comput. Electr. Eng.* **2018**, *66*, 274–287. [[CrossRef](#)]
2. Gulati, K.; Boddu, R.S.K.; Kapila, D.; Bangare, S.L.; Chandnani, N.; Saravanan, G. A review paper on wireless sensor network techniques in Internet of Things (IoT). *Mater. Today Proc.* **2021**. [[CrossRef](#)]
3. Rashid, B.; H, M. Rehmani. Applications of wireless sensor networks for urban areas: A survey. *J. Netw. Comput. Appl.* **2016**, *60*, 192–219. [[CrossRef](#)]
4. Adeel, A.; Gogate, M.; Farooq, S.; Ieracitano, C.; Dashtipour, K.; Larijani, H.; Hussain, A. A Survey on the Role of Wireless Sensor Networks and IoT in Disaster Management. In *Geological Disaster Monitoring Based on Sensor Networks*; Durrani, T.S., Wang, W., Forbes, S.M., Eds.; Springer: Singapore, 2019; pp. 57–66.
5. Nam, W.-H.; Kim, T.; Hong, E.-M.; Choi, J.-Y.; Kim, J.-T. A Wireless Sensor Network (WSN) application for irrigation facilities management based on Information and Communication Technologies (ICTs). *Comput. Electron. Agric.* **2017**, *143*, 185–192. [[CrossRef](#)]
6. de Souza, P.S.S.; Rubin, F.; Hohemberger, R.; Ferreto, T.C.; Lorenzon, A.F.; Luizelli, M.C.; Rossi, F.D. Detecting abnormal sensors via machine learning: An IoT farming WSN-based architecture case study. *Measurement* **2020**, *164*, 108042. [[CrossRef](#)]
7. Belghith, A.; S, M. Obaidat. Chapter 2—Wireless sensor networks applications to smart homes and cities. In *Smart Cities and Homes*; Obaidat, M.S., Nicopolitidis, P., Eds.; Morgan Kaufmann: Boston, MA, USA, 2016; pp. 17–40.
8. Silva, B.N.; Khan, M.; Han, K. Towards sustainable smart cities: A review of trends, architectures, components, and open challenges in smart cities. *Sustain. Cities Soc.* **2018**, *38*, 697–713. [[CrossRef](#)]
9. Shabbir, N.; Hassan, S.R. Routing Protocols for Wireless Sensor Networks (WSNs). In *Wireless Sensor Networks*; Sallis, P., Ed.; IntechOpen: Rijeka, Croatia, 2017.
10. Tan, K.; Wu, D.; Chan, A.J.; Mohapatra, P. Comparing simulation tools and experimental testbeds for wireless mesh networks. *Pervasive Mob. Comput.* **2011**, *7*, 434–448. [[CrossRef](#)]
11. Maamar, S. Evaluation of QoS parameters with RPL protocol in the Internet of Things. In Proceedings of the International Conference on Computing for Engineering and Sciences, Shanghai, China, 22–24 July 2017; pp. 86–91.
12. Lera, I.; Guerrero, C.; Juiz, C. YAFS: A Simulator for IoT Scenarios in Fog Computing. *IEEE Access* **2019**, *7*, 91745–91758. [[CrossRef](#)]

13. Barthel, D.; Vasseur, J.P.; Pister, K.; Kim, M.; Dejean, N. Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks. In *RFC 6551*; IETF: Fremont, CA, USA, 2012. [[CrossRef](#)]
14. Muñoz, A.; Gonzalez, J.; Maña, A. A Performance-Oriented Monitoring System for Security Properties in Cloud Computing Applications. *Comput. J.* **2012**, *55*, 979–994. [[CrossRef](#)]
15. Serrano, D.; Ruíz, J.F.; Muñoz, A.; Maña, A.; Armenteros, A.; Crespo, B.G.-N. Development of Applications Based on Security Patterns. In Proceedings of the 2009 Second International Conference on Dependability, Manchester, UK, 21–22 May 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 111–116. [[CrossRef](#)]
16. Osterlind, F.; Dunkels, A.; Eriksson, J.; Finne, N.; Voigt, T. Cross-Level Sensor Network Simulation with COOJA. In Proceedings of the 2006 31st IEEE Conference on Local Computer Networks, Tampa, FL, USA, 14–16 November 2006; pp. 641–648. [[CrossRef](#)]
17. Sarkar, A.; Murugan, T.S. Routing protocols for wireless sensor networks: What the literature says? *Alex. Eng. J.* **2016**, *55*, 3173–3183. [[CrossRef](#)]
18. Bokde, N.D.; Peshwe, P.D.; Gupta, A.; Kulat, K.D. RemoteWSN: A novel technique for remotely visualizing connectivity in WSN working on a weight based routing algorithm. In Proceedings of the 2015 National Conference on Recent Advances in Electronics Computer Engineering (RAECE), Roorkee, India, 13–15 February 2015; pp. 92–95. [[CrossRef](#)]
19. Qaqos, N.; Zeebaree, S.; Hussan, B. Opnet Based Performance Analysis and Comparison Among Different Physical Network Topologies. *Acad. J. Nawroz Univ.* **2018**, *7*, 48–54. [[CrossRef](#)]
20. Al-khatib, A.A.; Hassan, R. Performance Evaluation of AODV, DSDV, and DSR Routing Protocols in MANET Using NS-2 Simulator. In *Recent Trends in Information and Communication Technology*; Springer: Cham, Switzerland, 2017; pp. 276–284.
21. Bukhari, S.H.R.; Siraj, S.; Rehmani, M.H. NS-2 based simulation framework for cognitive radio sensor networks. *Wirel. Netw.* **2016**, *24*, 1543–1559. [[CrossRef](#)]
22. Baidya, S.; Shaikh, Z.; Levorato, M. FlyNetSim: An Open Source Synchronized UAV Network Simulator Based on Ns-3 and Ardupilot. In Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, Montreal, QC, Canada, 28 October 2018; pp. 37–45. [[CrossRef](#)]
23. Hayamizu, Y.; Matsuzono, K.; Asaeda, H. CeforeSim: Cefore Compliant NS-3-Based Network Simulator. In Proceedings of the 2019 IEEE 27th International Conference on Network Protocols (ICNP), Chicago, IL, USA, 8–10 October 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–2.
24. Varga, A.; Hornig, R. An Overview of the OMNeT++ Simulation Environment. In Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops, Marseille, France, 3–7 March 2008; ICST: Kolkata, India, 2008.
25. Varga, A. A practical introduction to the OMNeT++ simulation framework. In *Recent Advances in Network Simulation*; Springer: Berlin, Germany, 2019; pp. 3–51.
26. Udugama, A.; Förster, A.; Dede, J.; Kuppusamy, V.; Muslim, A.B. Opportunistic Networking Protocol Simulator for OMNeT++. *arXiv* **2017**, arXiv:1709.02210.
27. Sliwa, B.; Ide, C.; Wietfeld, C. An OMNeT++ based Framework for Mobility-aware Routing in Mobile Robotic Networks. *arXiv* **2016**, arXiv:1609.05351.
28. Levis, P.; Lee, N.; Welsh, M.; Culler, D. TOSSIM: Accurate and scalable simulation of entire TinyOS applications. In Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, Los Angeles, CA, USA, 5–7 November 2003; pp. 126–137.
29. Al-Roubaiey, A.; Sheltami, T.; Mahmoud, A.; Yasar, A. EATDDS: Energy-aware middleware for wireless sensor and actuator networks. *Futur. Gener. Comput. Syst.* **2019**, *96*, 196–206. [[CrossRef](#)]
30. Mora-Merchan, J.M.; Larios, D.F.; Barbancho, J.; Molina, F.J.; Sevillano, J.L.; León, C. mTOSSIM: A simulator that estimates battery lifetime in wireless sensor networks. *Simul. Model. Pract. Theory* **2013**, *31*, 39–51. [[CrossRef](#)]
31. Sabor, N.; Sasaki, S.; Abo-Zahhad, M.; Ahmed, S.M. A Graphical-based educational simulation tool for Wireless Sensor Networks. *Simul. Model. Pract. Theory* **2016**, *69*, 55–79. [[CrossRef](#)]
32. Youssef, A.; Maheswaran, M.; Youssef, L. Wireless GINI: An educational platform for hosting virtual wireless networks. *Softw. Pract. Exp.* **2017**, *47*, 21–59. [[CrossRef](#)]
33. Goyal, T.; Singh, A.; Agrawal, A. Cloudsim: Simulator for cloud computing infrastructure and modeling. *Procedia Eng.* **2012**, *38*, 3566–3572. [[CrossRef](#)]
34. Son, J.; He, T.; Buyya, R. {CloudSimSDN}-{NFV}: Modeling and simulation of network function virtualization and service function chaining in edge computing environments. *Softw. Pract. Exp.* **2019**, *49*, 1748–1764. [[CrossRef](#)]
35. Gupta, H.; Vahid Dastjerdi, A.; Ghosh, S.K.; Buyya, R. {iFogSim}: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. *Softw. Pract. Exp.* **2017**, *47*, 1275–1296. [[CrossRef](#)]
36. Contiki, O.S. Contiki: The Open Source Operating System for the Internet of Things. *Wirel. Netw.* **2018**, *55*, 1543–1559. [[CrossRef](#)]
37. Sitanayah, L.; Sreenan, C.J.; Fedor, S. A Cooja-based tool for coverage and lifetime evaluation in an in-building sensor network. *J. Sens. Actuator Netw.* **2016**, *5*, 4. [[CrossRef](#)]
38. Ferracuti, F.; Freddi, A.; Monteriù, A.; Prist, M. An Integrated Simulation Module for Cyber-Physical Automation Systems. *Sensors* **2016**, *16*, 645. [[CrossRef](#)] [[PubMed](#)]

39. Bumb, A.; Iancu, B.; Cebuc, E. Extending Cooja simulator with real weather and soil data. In Proceedings of the 2018 17th RoEduNet Conference: Networking in Education and Research (RoEduNet), Cluj-Napoca, Romania, 6–8 September 2018; pp. 1–5. [[CrossRef](#)]
40. Aminian, B.; Araújo, J.; Johansson, M.; Johansson, K.H. GISOO: A virtual testbed for wireless cyber-physical systems. In Proceedings of the IECON 2013-39th Annual Conference of the IEEE Industrial Electronics Society, Vienna, Austria, 10–13 November 2013; pp. 5588–5593.
41. Cheong, E.; Lee, E.A.; Zhao, Y. Viptos: A Graphical Development and Simulation Environment for TinyOS-Based Wireless Sensor Networks. February 2006. Available online: <http://www2.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-15.html> (accessed on 2 November 2005).
42. Baldwin, P.; Kohli, S.; Lee, E.A.; Liu, X.; Zhao, Y.; Ee, C.T.; Zhou, R. Visualsense: Visual modeling for wireless and sensor network systems. *Tech. Memo. UCB/ERL* **2005**, 5.
43. Ptolemaeus, C. *System Design, Modeling, and Simulation: Using Ptolemy II*; Ptolemy.org: Berkeley, CA, USA, 2014; Volume 1.
44. Cheong, E. PtinyOS: Simulating TinyOS in Ptolemy II. Ptolemy.org. 2004. Available online: <http://ptolemy.org/books/Systems> (accessed on 18 August 2021).
45. Shu, L.; Hauswirth, M.; Chao, H.-C.; Chen, M.; Zhang, Y. NetTopo: A framework of simulation and visualization for wireless sensor networks. *Ad. Hoc. Netw.* **2011**, 9, 799–820. [[CrossRef](#)]
46. Ma, L.; Wang, L.; Shu, L.; Zhao, J.; Li, S.; Yuan, Z.; Ding, N. NetViewer: A Universal Visualization Tool for Wireless Sensor Networks. In Proceedings of the 2010 IEEE Global Telecommunications Conference GLOBECOM 2010, Miami, FL, USA, 6–10 December 2010; pp. 1–5. [[CrossRef](#)]
47. Castillo, J.A.; Ortiz, A.M.; López, V.; Olivates, T.; Orozco-Barbosa, L. WiseObserver: A Real Experience with Wireless Sensor Networks. In Proceedings of the 3rd ACM Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks, New York, NY, USA, 31 October 2008; pp. 23–26. [[CrossRef](#)]
48. Wang, Z.; Zhang, L.; Zheng, Z.; Wang, J. Energy balancing RPL protocol with multipath for wireless sensor networks. *Peer-Peer Netw. Appl.* **2018**, 11, 1085–1100. [[CrossRef](#)]
49. Summerfield, M. *Rapid GUI Programming with Python and Qt: The Definitive Guide to PyQt Programming (Paperback)*; Pearson Education: London, UK, 2007.
50. Bucanek, J. Model-view-controller pattern. In *Learn Objective-C for Java Developers*; Apress: New York, NY, USA, 2009; pp. 353–402.
51. Gurgens, S.; Rudolph, C.; Mana, A.; Munoz, A. Facilitating the Use of TPM Technologies through S&D Patterns. In Proceedings of the 18th International Workshop on Database and Expert Systems Applications (DEXA 2007), Regensburg, Germany, 3–7 September 2007; pp. 765–769. [[CrossRef](#)]
52. Muñoz, A.; Sánchez-Cid, F.; el Khoury, P.; Compagna, L. {XACML} as a Security and Dependability Pattern for Access Control in {AmI} environments. In *Developing Ambient Intelligence*; Springer: Paris, France, 2008; pp. 143–155.
53. Sales, F.O.; Marante, Y.; Vieira, A.B.; Silva, E.F. Energy Consumption Evaluation of a Routing Protocol for Low-Power and Lossy Networks in Mesh Scenarios for Precision Agriculture. *Sensors* **2020**, 20, 3814. [[CrossRef](#)]
54. Avila, K.; Jabba, D.; Gomez, J. Security Aspects for Rpl-Based Protocols: A Systematic Review in IoT. *Appl. Sci.* **2020**, 10, 6472. [[CrossRef](#)]
55. Winter, T.; Thubert, P.; Brandt, A.; Hui, J.W.; Kelsey, R.; Levis, P.; Alexander, R.K. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. *RFC* **2012**, 6550, 1–157. [[CrossRef](#)]