6-2-2022

Simulating Distributed Wireless Sensor Networks for Edge-AI

Ambar Prajapati

SIMULATING DISTRIBUTED WIRELESS SENSOR NETWORKS FOR EDGE-AI

by
Ambar Prajapati

A Thesis
Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science

Major: Computer Engineering

The University of Memphis
May 2022

ACKNOWLEDGEMENT

I wish to express my honest regards and warm appreciation to my advisor Dr. Bonny Banerjee who made this work possible and contributed much to helping me shape and reshape this valuable piece of work. His guidance and advice carried me through the efforts across all stages of the research.

Thanks to Dr. Lau and Dr. Robinson for serving on the thesis defense committee. Thank you for letting me defend the thesis and for your valuable feedback.

Thanks to Dr. Lau for his valuable feedback earlier when we faced a design challenge on this thesis. The credit also goes to Dr. Robinson for being my first advisor who introduced me to the master's degree at UofM.

I wish also to thank my wife and sons for their continued support and understanding while working on the thesis and my master's degree. Your prayers let me sustain the efforts and keep moving.

Finally, I thank Almighty for letting me through all the challenges. I keep feeling showers of Your blessing every day.

ABSTRACT

This study presents the simulations for distributed wireless sensor networks (WSNs) of autonomous mobile nodes that communicate intelligently, with or without a central/root node, as is desired in Edge Artificial Intelligence (Edge-AI).

We harness the high-resolution and multidimensional sensing characteristics of IEEE 802.15.4 standard and Routing Protocol for Low-Power and Lossy Networks (RPL) to implement dynamic, asynchronous, event-driven, targeted communication in distributed WSNs in a simulator.

We use the chosen Contiki-NG/Cooja to simulate two WSNs with and without a central node. The two WSN simulations are assessed on the network Quality of Service (QoS) parameters such as throughput, network lifetime, power consumption, and packet delivery ratio. The simulation outputs show that the sensor nodes at the edge communicate successfully with the specific targets responding to particular events in an autonomous and asynchronous manner. However, the performance is seen slightly degraded in the RPL WSN network with a central node.

This work shows how to simulate distributed WSNs using the Cooja simulator, with or without a central node, for communication among sensors relevant to Edge-AI applications, such as visual surveillance, monitoring in assisted living facilities, intelligent transportation, connected vehicles, automated factory floors, immersive media experience, etc.

Contents

Introduction

1.1 Motivation

Edge computing involves data processing and analysis at the edge of a network, where actual data generation and collection happen. The Edge AI thus necessitates executing AI algorithms locally on an edge device or a server near the edge device. The devices should make un-assisted, uninterrupted autonomous decisions within a fraction of a millisecond without connecting to the Internet or the Cloud.

The design aims to create distributed wireless sensor networks (WSNs) consisting of edge devices that talk to their peers directly, take decisions on their own in a dynamic, asynchronous environment. The devices thus perform intelligent, independent communication within their constraints to save power and resources during Edge Computing.

As a simplified example of insightful predictive analysis and actionable decision-making in real-time, let us consider three rooms in a building, each fitted with a temperature sensor to register hourly local room temperature. After recording, each sensor validates its measured value against a predefined high/low-temperature threshold. On crossing the threshold, the sensor compares its reading with the neighboring sensors. If its measured value does not align with the other sensors, there is a likelihood that the sensor may have started malfunctioning.

However, in the other case, when the measured value crosses the threshold and matches with similar spikes in neighboring sensors, there is another likelihood of abnormal hot or cold temperature build-up in that area. The sensor then communicates with the expert sensors for follow-up actions such as sending alerts for possible dangers to a central command station, lighting the bulbs or ringing the alarms, running water showers in case of fire, and powering on the air-conditioners for temperature adjustment, etc.

The building represents a dynamic neighborhood where the measuring sensors can be installed anew, removed, or moved to random locations.

Thus,

1. The sensors communicate when they need to, e.g., when crossing the threshold or during the spikes (asynchronous event-triggered communication).
2. They talk to expert sensors for a follow-up action (targeted communication).
3. The sensors belong to a network where they leave or join the network randomly (dynamic network).

In this work, we will use an existing simulator software to simulate a working system for practical implementation.

1.2 Contributions

Towards the aim of generating continuously integrated real-time data and actionable insights and decisions based on artificial intelligence (AI) or machine learning (ML) processing at the cloud-agnostic constrained devices over Edge AI platforms, the contributions of this work are:

1. Comparing existing simulators and identifying one that is affordable and can
    - simulate asynchronous, event-triggered, and targeted communication in a dynamic sensor network,
    - serve as an emulator with an actual edge device's hardware/software feature to facilitate real-world deployment with minimal effort.

   Remark: *Please note that the study to compare the simulators was carried out during Spring 2021 as part of EECE 7992 Independent Study II.*
2. Implementing point-to-point communication for edge computing in scalable distributed wireless sensor networks with varying network topologies.
3. Demonstrating actionable decision making based on simple conditional logic at sensor level in distributed WSN using
    - high-resolution and multidimensional IEEE 802.15.4 standard,
    - Routing Protocol for Low-Power and Lossy Networks (RPL).

2 Simulation Models and Methods

2.1 Problem Definition

Design and implement a distributed wireless sensor network (WSN) or Internet of Things (IoT) network capable of targeted, asynchronous, event-triggered, dynamic nodal communication with no central storage and no centralized decision-making.

Thus, we aim to build a distributed WSN where:

- Each node (or sensor or thing) decides when and whom to communicate. Hence, when triggered by an asynchronous event, the node communicates with a target node running a decision matrix locally. The source node thus exhibits complete autonomy based on its unique data, knowledge, and capabilities.
- The nodes are stationary or moving. Hence, the network topology is expected to change with nodal mobility.
- The independent decision making by a node consists of a few simple if-then statements like below, representing unique conditions, unique messages by unique nodes:

```
if <condition>,
    Send <message> to <node>
```

2.2 Simulator Selection

To demonstrate the capabilities of our chosen networks on Edge AI, we needed an IoT simulator to emulate real-world scenarios on distributed WSNs. Earlier works available in the literature on simulator comparison do not evaluate the latest version of simulators on network attributes of our choice, namely, event-triggered, targeted, asynchronous, and dynamic communication. So, we present here our evaluation and comparison of a few popular IoT simulators, namely Contiki/Cooja, OMNeT++/Castalia, NS-3, and MATLAB-Simulink. Another simulator named NuvIoT had to be excluded as it frequently errored at the time of this research on its MS-Windows and web versions.

- **OMNeT++**: The OMNeT++ has an event network simulation framework suitable for modeling entities communicating by exchanging messages. The nodes generate realistic packets based on a real-life scenario.
- **NS-3**: The NS-3 event network simulator executes events in sequential time order at a specified simulation time. The NS-3 generates consistent, reproducible results for modeling the internet protocols on real network cards.
- **MATLAB – Simulink**: Matlab-Simulink allows creating digital twins and developing data-driven and physics-based models for simulating the end nodes, modeled in continuous, sampled, or a hybrid time for connected things.
- **Contiki-NG/Cooja:** With Cooja, large and small networks of Contiki nodes can be simulated. Contiki OS allows precise inspection of the nodes and directly emulates their behavior at the hardware level.

The detailed feature comparison for the above simulators is shown in Table 1.

2.2.1 Limitations of the simulators not selected for this study

Referring to the recent literature, the OMNeT++, NS-3, and MATLAB/Simulink were opted out from further studies due to their limitations listed below.

[A] NS-3

- A limited number of contributed source code [31], implying lesser interest in the development community
- Lesser support for protocols and devices using pre-built models [31]
- Higher CPU usage impacting performance [31]
- The NS-3 simulation framework cannot be integrated with an actual hardware platform [35]

[B] OMNeT++

- The Castalia/OMNeT++ kernel is single-threaded. As OMNeT++ only uses one core, so a higher single-core performance causes higher execution speeds and fully loading of only one core of the CPU [32]
- OMNeT++ does not support multi-hop communication which strongly depends on real-time data [32]

- OMNeT++ is free only for academic and non-profit use [33]
- The INET Framework behind OMNeTT++ is not specialized in mobile and wireless networks but has some support for it [34]
- The OMNeT++ simulation framework cannot be integrated with an actual hardware platform [35]

[C] MATLAB/Simulink

- Max number of permitted nodes for simulation are only 100 [35]
- Affordability issue

2.2.2 The rationale for choosing Contiki-NG/Cooja as our simulator

"*Cooja is an emulator – a hardware or software system that makes the host system - a complete replication of the guest system, right down to being binary compatible with the emulated system's inputs and outputs. The code to be executed by the node is the exact same firmware uploaded to physical nodes*" [47].

"*Cooja Framework, due to its use of embedded software to perform cycle-exact emulation of devices— can run any program designed for one of the emulated architectures. Many published articles in the domain of WSNs, including very recent publications, include simulations made with the Cooja/MSPSim framework. Cooja has the ability to develop and debug WSN-related software much more easily thanks to its emulation features.*" [46].

Thus, we observe that the Contiki NG/Cooja has several advantages over other simulators for edge devices. Its properties listed below allow the simulation of event-triggered, targeted, asynchronous, dynamic communication along with energy efficiency, efficient bandwidth usage, efficient coverage, and connectivity for use in distributed WSN.

- Preemptive multithreading, proto-thread based concurrent programming, event-driven kernel
- Wide range of supported protocols, propagation models
- Emulated code directly portable on sensor hardware platforms
- Open source along with substantial industrial and development support
- Supported simulation for 170+ nodes

The following sections describe our simulations developed in Contiki NG/Cooja.

Feature	**Contiki**	**Castalia**	**NS-3**	**MATLAB-Simulink**
Latest version	Contiki-NG Version 4.6	Version 3.3	Version NS-3.7	Version R2021a
Programming language	C language, Optional GUI	C++, OMNeT++ NED language	C++/Python, No GUI	MATLAB multi-paradigm and Simulink proprietary graphical programming languages
Required library or software	Native Contiki OS libraries	OMNeT++ component-based C++ simulation library and framework	Standard C++ library: GPLv2	MATLAB coder, Simulink
OS/Middleware	Contiki OS	Any OS with a modern C++ compiler. But Simulation IDE can run only on Windows, Linux, or macOS	Linux Cygwin, osX	Windows, macOS, and Linux
License type	Open-sourced under a BSD-style license.	Academic Public License / Commercial License for OMNeT++ for commercial usage	Free, GNU GPLv2 license	Commercial license required for usage
Simulator type	Sensor simulator called Cooja, which simulates Contiki nodes	OMNeT++ platform based object-oriented modular discrete event network simulation framework	discrete-event network simulator for Internet systems	Graphical programming environment – Simulink tightly integrated with MATLAB
Simulation library features	The simulation platform provides users with a single, integrated GUI environment in which all tasks are carried out.	Discrete event simulation, scheduling events, sending and receiving messages, channel operation, finite state machines, dynamic module creation, signals, logging, random number generation, queues, topology discovery and routing support, statistics and result collection	*"NS-3 is designed as a set of libraries that can be combined together and also with other external software libraries. Several external animators and data analysis and visualization tools can be used with ns-3."* [6]	Discrete-event simulation with Simulink® provides capabilities for analyzing and optimizing event-driven communications and operations using hybrid system models, agent-based models, and state charts. Its primary graphical block interface is a diagramming tool and a customizable block library.

Uniqueness	"*Contiki is a multitasking operating system, specially designed for microcontrollers with small amount of memory (35KB of ROM and around 3K of RAM)*"[9]. Its in-built TCP/IP stack provides lightweight preemptive scheduling over event-driven kernel	"*Researchers and developers use Castalia to test their distributed algorithms and/or protocols in a realistic wireless channel and radio models*" [10] behavior especially relating to access of the radio. Used as a simulator for Wireless Sensor Networks (WSN), Body Area Networks (BAN), and generally networks of low-power embedded devices.	"*Open, extensible network simulation platform, for networking research and education. Some of the reasons to use ns-3 include performing studies that are more difficult or not possible to perform with real systems, to study system behavior in a highly controlled, reproducible* environment" [11], for modeling the Interne t protocols and to learn about how networks work. ns-3 can also be used to model non-Internet-based systems.	Easy to use, wide adoption, mature product, great visualization for complex systems. Matlab's simulations of physical structures can almost automate entire cycles in product delivery.
Event-driven programming	Uses Event-Driven Kernel based on protothreads. A protothread is a concurrent programming mechanism "*that shares features of both multithreading and event-driven programming to attain a low memory overhead*" [12]	Uses OMNeT++ event-driven simulation engine	Event-driven simulation core and object framework. *"Conceptually, the simulator keeps track of a number of events that are scheduled to execute at a specified simulation time. The job of the simulator is to execute the events in sequential time order. Once the completion of an event occurs, the simulator will move to the next event (or will exit if there are no more events in the event queue)"* [13]	SimEvents is used to add a library of graphical building blocks for modeling queuing systems to the Simulink environment and to add an event-based simulation engine to the time-based simulation engine in Simulink.
Preemptive multithreading	Contiki provides preemptive multithreading as an application library that runs on top of the event-based kernel. Preemptive	cooperative multitasking or non-preemptive threads - The Threads are scheduled non-preemptively	NS-3 provides a non-preemptive scheduler via (direct-code execution) DCE Manager. This scheduler allows a 'synchronous' programming	A custom or pred-defined architecture can be chosen for models configured for concurrent execution. Data and task

	multithreading can be provided on a per-process basis. Contiki processes run in the cooperative context, whereas interrupts and real-time timers run in the preemptive context.		style where functions can block until certain conditions are verified as opposed to event-driven programming.	parallelism including pipelines can be implemented in Simulink.
Supported Protocols and Communication Stacks	Implements IPv6 and IPv4 stacks, along with the recent low-power wireless standards : 6LoWPAN, RPL, CoAP, TSCH, Nullnet layer	"*INET Framework is an open-source model library for the OMNeT++ simulation environment. INET supports a wide class of communication networks, including wired, wireless, mobile, ad hoc and sensor networks. It contains models for the Internet stack (TCP, UDP, IPv4, IPv6, OSPF, BGP, etc.), link-layer protocols (Ethernet, PPP, IEEE 802.11, various sensor MAC protocols, etc.), refined support for the wireless physical layer, MANET routing protocols, DiffServ, MPLS with LDP and RSVP-TE signaling, several application models, and many other protocols and components. It also provides support for node mobility, advanced visualization, network emulation, and more*" [14]	IPv4, UDP, and TCP stack and support for NSC (integrating Linux and BSD TCP/IP network stacks).	Simulink supports two communication mechanisms [1]- XCP, the Universal Measurement , and Calibration Protocol [2]- TCP/IP and serial (RS-232)

Supported Propagation Models	- Unit Disk Graph Medium (UDGM) - Distance loss UDGM - Directed Graph Radio Medium (DGRM) - Multipath Ray-tracer Medium (MRM)	Default is Free Space model Extended by paper-10 these propagation models- Two Ray Ground and Log-Normal Shadowing, Nakagami model, probabilistic propagation models	Free Space, Two Ray Ground and Log-Normal Shadowing, Nakagami Model. A set of solid 802.11 MAC and PHY models	MATLAB Simulink supports various propagation models, e.g., Atmospheric, Empirical, Terrain and Ray Tracing methods – image and SBR (shooting and bouncing rays)
Mesh networking	Contiki supports more advanced functionality like mesh networking, an important capability for self-forming and self-healing large networks with many nodes.	OMNeT++ supports simulation of 802.11 Wireless Mesh Networks (WMNs).	NS-3 supports mesh networking as IEEE 802.11s Mesh Networking Model	Mesh Networking supported e.g. network layer flooding in a Bluetooth® mesh network using Communication Toolbox Library™ for the Bluetooth® Protocol.
Real-time emulation features	The simulation code generated can be deployed mostly as-is to work on real physical systems.	Does not meet hard real-time requirements – rather focuses on soft and firm real time	*"To integrate with real network stacks and emit/consume packets, NS-3 uses a real-time scheduler to lock the simulation clock with the hardware clock."* [15] The real-time scheduler causes the progression of the simulation clock to occur synchronously to some external time base.	The simulation code generated can be deployed mostly as-is to work on real physical systems.

Table 1. Comparison of simulators' features. **Remark:** *Please note that the study to compare the simulators was carried out during Spring 2021 as part of EECE 7992 Independent Study II.*

2.3 Simulation of a distributed WSN without a central/root node

This section implements a distributed WSN, an autonomous collection of mobile nodes that communicate over wireless links without a central node. Revisiting our requirements, we need the nodes to be mobile and independently execute logical decisions, network organization, and message delivery.

The left-most layers shown in Figure 1(a) represent the networking framework of the Open System Interconnections (OSI) model that conceptualizes communication among systems. The media access control (MAC) sublayer of the data link layer has access to the wireless medium for data transmission. It provides an abstraction of the physical layer to the layers above it in the OSI framework [48]. The MAC layer encapsulates IEEE 802.15.4 standard for low-rate wireless personal area networks (LR-WPANs). This standard provides ubiquitous communication among nearby devices with little to no underlying infrastructure.

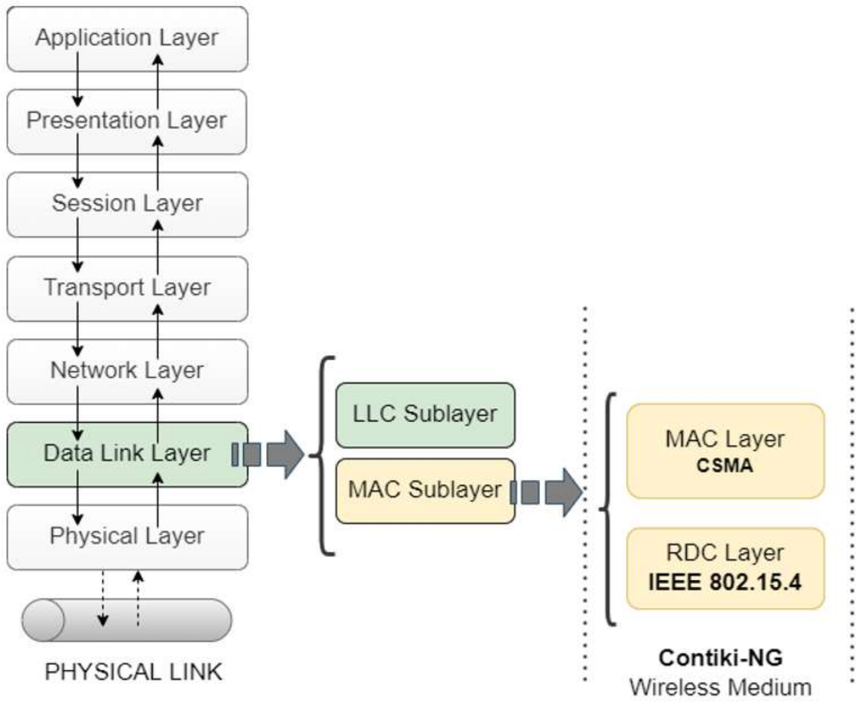


Figure 1(a). Open System Interconnections (OSI) Data-link layer mapped to Contiki-NG

The MAC layer governs the transmission and reception of data packets. ContikiMAC, the default MAC protocol in ContikiOS, uses Carrier Sensing for detecting medium activity. The Radio Duty Cycle RDC protocol performs asynchronous discovery using low-power probing and low-power listening (LPL) [59]. The mechanism works so that the entire data frames are sent repeatedly until acknowledged by the receiver. The nodes detect the emerging links in the neighborhood using their constantly-on radios that operate at low-duty cycles to preserve power and maximize lifetime.

With short wake-up periods and an efficient transmission mechanism, the MAC/RDC layers yield a better Packet Delivery Ratio (PDR) and work well in practice for dynamic traffic [55] [58].

On the right of Figure 1(a), we show how the MAC Sublayer maps with the Contiki network protocol stack shown in Figure-1(b). The MAC layer of Contiki-NG, version 4.0 onwards, combines MAC and the Radio Duty Cycle (RDC) layers [16]. The RDC layer controls the wake-up and sleep cycles of a node to preserve energy during data transmission.

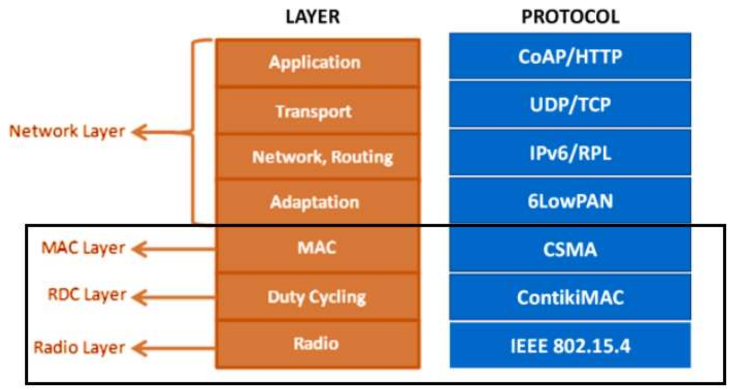


Figure 1(b). Contiki protocol stack. Adopted from [4].

With version 4.0, a new layer called Nullnet is introduced in the Contiki-NG network stack [16]. The Nullnet network layer is a minimal layer helpful for lower-layer testing and non-IPv6 scenarios [38]. It relays the data packets unmodified up/down the Contiki network stack. The data packets from the sender node travel via MAC and NullNet layers up to the receiver node.

The network layer in Contiki OS is accessed via the global variable NETSTACK_NETWORK defined in compilation time at *core/net/netstack.h* [55][56] and the NullNet layer is accessed via *os/net/nullnet/nullnet.h* [57].

Thus, the dual-stack of the MAC and Nullnet layers in Contiki-NG ensures a distributed WSN without utilizing a central node or any intermediate device, access point, router, modem, gateway, cloud from the upper layers of the stack. Each node in a WSN is identified by its Extended Unique Identifier (EUI)-48-bit address [39][40], which is hard-wired into the sensor's transceiver for data transmission within a wireless (Wi-Fi, Zigbee, Bluetooth) segment. The lower dual-stack facilitates a unicast, any-node to any-node communication among nodes based on their EUI addresses.

2.3.1 Neighbor Discovery Mechanism Without a Central/Root Node

In the case of WSN with no central node: For discovering the MAC addresses, the ContikiOS caches the node addresses by default by enabling the global variable UIP_ND6_AUTOFILL_NBR_CACHE. The caching lets the node derive the MAC address from the EUI-64 contained in the IPv6 address [64]. However, if the neighbors are not maintained in the cache, the ContikiOS assumes auto-configuration and enables UIP_ND6_SEND_NS instead. In this case, it runs IPv6 Neighbor Discovery and derives the link-layer address of the neighbors from the link-local IPv6. Thus, it adds the neighbors regardless of their reachability and liveness. [65]

Simplifying further, in ContikiOS, a node consists of an Internet Protocol (IP) interface attached to the IEEE 802.15.4 medium link. During the entry of a node into the network at its system start-up, the node undergoes autoconfiguration and ensues an IPv6-based neighbor discovery process described below [62][66][67][68].

The node generates a tentative link-local address for the interface based on its MAC address. However, before self-assigning this address, the node multicasts a "*neighbor solicitation"* message to verify if its prospective link-local address is unique on the link.

1- If the address is already present on the link, a "*neighbor advertisement"* message is returned in the response. The autoconfiguration stops in this case, and manual intervention is required.
2- If the address is not in use nor is solicited by any of the neighbors, the link-local address is assigned to the interface, and the node gets attached to the link. The nodes attached on the same link become neighbors and cache the detected link-layer addresses for subsequent use.
3- A node is called reachable when the one-way packets forwarded to it by its neighbors reach its IP interface. The link-attached reachable neighbors can thus communicate or exchange data packets.

During exit and re-entry, if the node undergoes a change in its link-layer address, it multicasts a neighbor advertisement to all its neighbors [69]. It allows the neighbors to update their address cache with the modified link-layer address of the node.

2.3.2 Radio Propagation Models

2.3.2.1 Unit Disk Graph Medium (UDGM) distance-loss radio propagation

Unit Disk Graph Medium (UDGM) distance-loss radio propagation model in Cooja models the transmission range surrounding a unit disk with the transmitting node as the center. All the receivers within the disk have the same probability of successfully receiving a data packet. The UDGM uses distance as the criteria to determine if the two nodes can communicate with each other. It creates a directed graph of nodes within transmission distance of each other and displays "*the percentage of packets that will not be corrupted on transmission (TX) and reception (RX)"* [50]. It models interferences and the packets "*transmitted with a TX success ratio and received with RX success ratio, offering a higher sense of reality for the simulation*" [49]. Beyond the transmission disk around a node center, the UDGM-distance loss model draws another disk of interference, in which the packets from a node can interfere with those from the others. The data packets are delivered only when there is no interference during transmission.

2.3.2.2 Multi-path Ray-tracer Medium (MRM) radio propagation

Cooja provides yet another packet-based Multi-path Ray-tracer Medium (MRM) radio propagation model, which uses a ray-tracing technique in 2D space. The MRM environment in Cooja allows simulating signal-to-ratio (SNR) based reception, background-noise mean and variance, capture effects, path-rays, refraction, diffraction, reflection, and obstacle attenuation. The parameters for transmitter, receiver, ray-tracer, and obstacles can be configured via the MRM settings in Cooja.

The analytical approach of the MRM radio propagation model approximates obstacles as attenuators of the signal strength. The model accounts for the refractions, reflections, and diffractions to render a much realistic transmission range of radio signals. [70]

We generated a WSN (with no central node) as in Figure 2(a), in MRM environment in Cooja that shows the radio antennas of five nodes including an obstacle. The nodes were placed at distances ranging from 200 to 800 meters. An obstacle, shown as the rectangular walls was placed between the nodes 2 and 5.

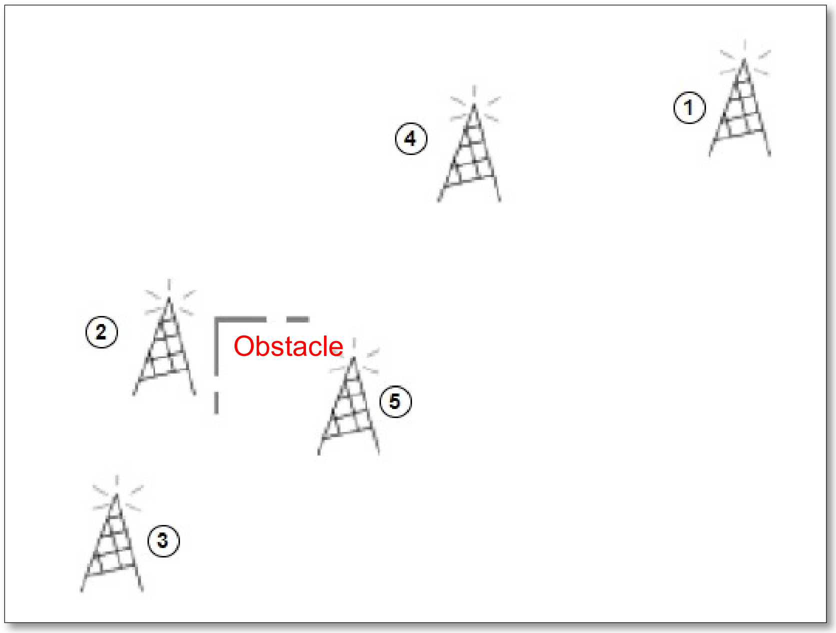


Figure 2(a). An obstacle placed between the nodes 2 and 5 in the MRM environment

Figure 2(b) shows the same five nodes in the MRM environment with their positions in 2D space and the probability of reception, signal strength from node 5.

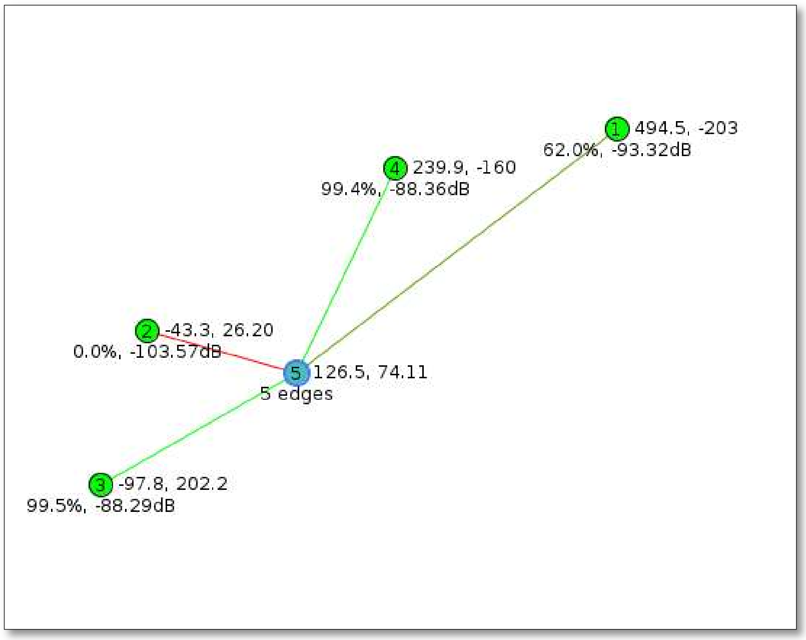


Figure 2(b). Impact of obstacle on the probability of packet reception as seen from node 5 in the MRM environment

From Figure 2(b), we observe that node 2 has a 0% probability of reception for the packets from node 5 due to an obstacle between the two.

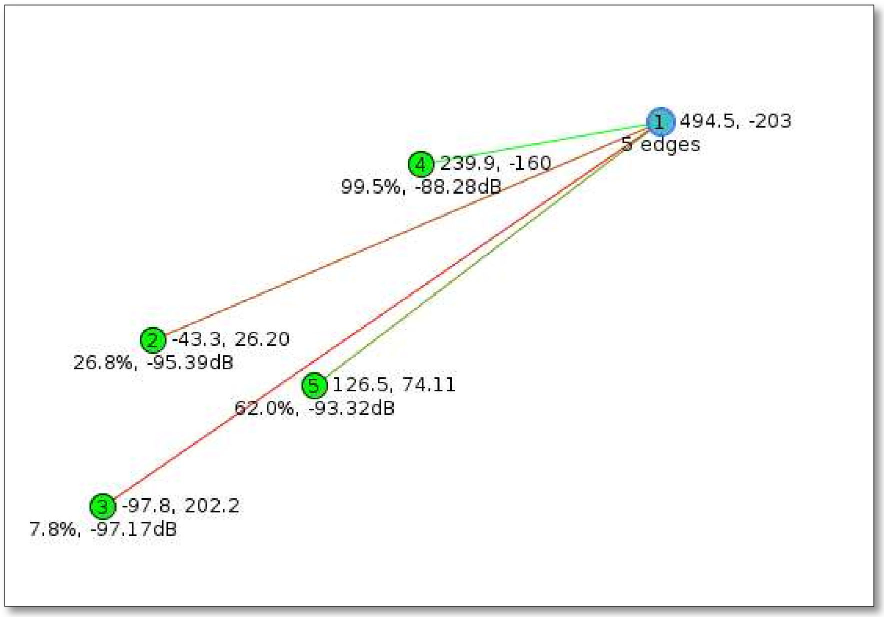
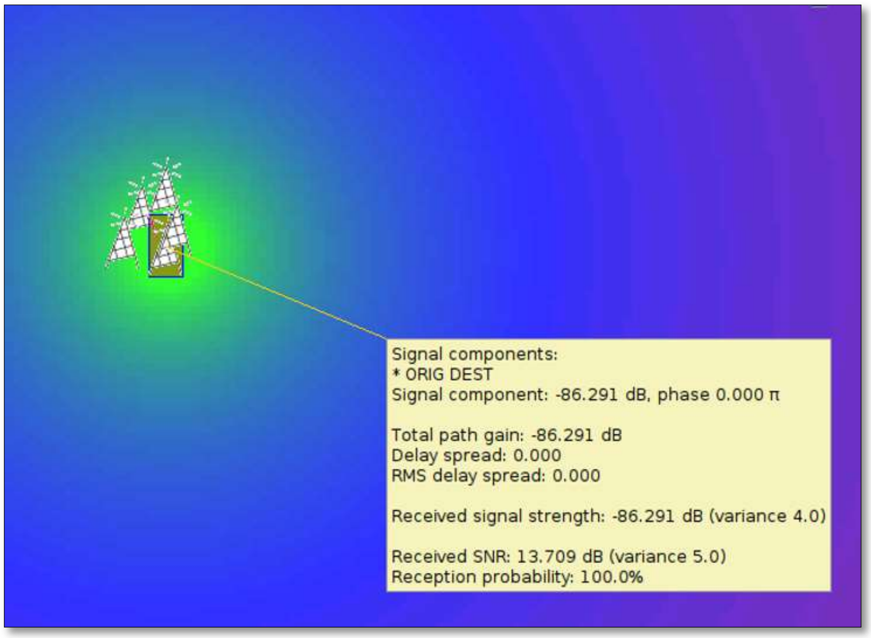


Figure 2(c). Impact of distance on the probability of packet reception as seen from node 1 in the MRM environment

From Figure 2(c), we observe that as we move farther away from node 1 towards nodes 5, 2, and 3, the probability of reception drops to just 7.8%

In Figures 2(d) and 2(e), we show the color-painted MRM radio environment for visualization of the signal strength around node 5 at two different distances (measured by the length of the ray).

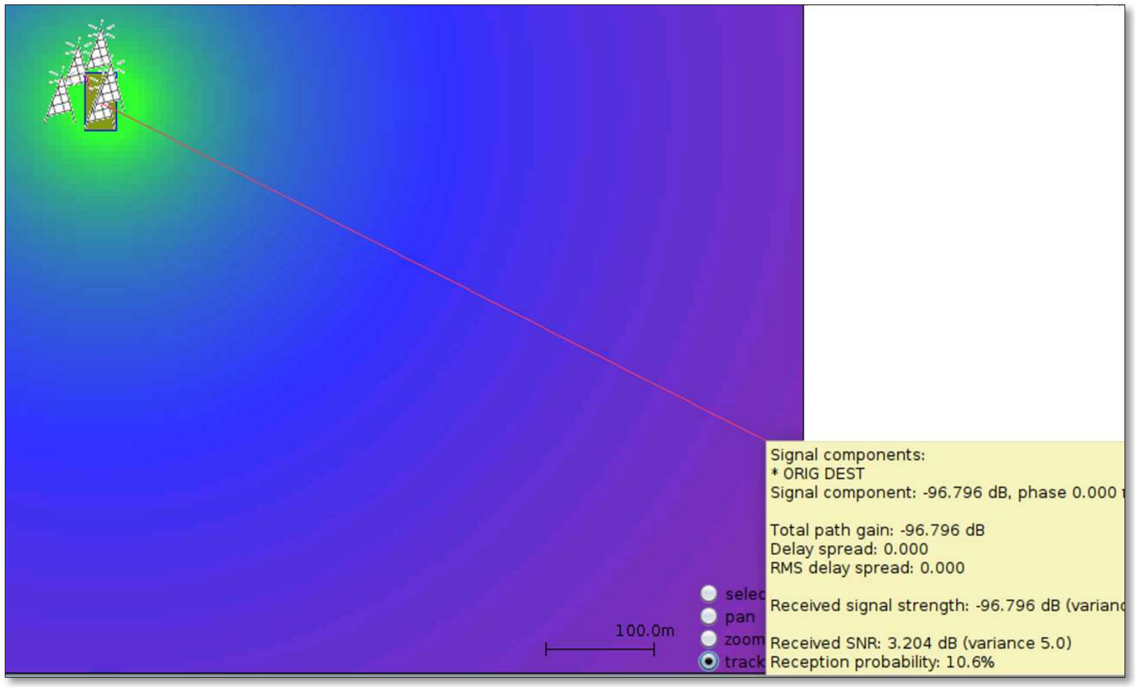

Signal components:
* ORIG DEST
Signal component: -86.291 dB, phase 0.000 π

Total path gain: -86.291 dB
Delay spread: 0.000
RMS delay spread: 0.000

Received signal strength: -86.291 dB (variance 4.0)

Received SNR: 13.709 dB (variance 5.0)
Reception probability: 100.0%

Figure 2(d). MRM signal reception at a distance of about 200 m away from the node



Signal components:
* ORIG DEST
Signal component: -96.796 dB, phase 0.000

Total path gain: -96.796 dB
Delay spread: 0.000
RMS delay spread: 0.000

Received signal strength: -96.796 dB (varianc

Received SNR: 3.204 dB (variance 5.0)
Reception probability: 10.6%

Figure 2(e). MRM signal reception at a distance of about 700 m away from the node

We tracked rays from a node at different distances in the MRM environment to inspect the signal strength, signal-to-noise ratio, path gain, and reception probability when moving away from the node. The observations were recorded in Table A.

Distance (in meters)	100	200	300	400	500	600	700
Received Signal strength (in dB) at variance 4.0	-80.113	-86.223	-90.100	-92.361	-94.197	-96.091	-96.796
Total path gain	-80.113	-86.223	-90.100	-92.361	-94.197	-96.091	-96.796
Received SNR (in dB) at variance 5.0	19.887	13.895	9.909	7.856	5.851	3.921	3.204
Reception Probability	100%	100%	96.2%	81%	47.1%	17.6%	10.6%

Table A. Signal quality at various distances from a node
in MRM radio environment in Cooja for a WSN without a central node

From Table A, we observe that the SNR and reception probability begin to drop at distances away from a node.

To assess the impact of obstacle, distances, interference, refraction, and diffraction on the network Quality of Service (QoS) parameters (i.e., network throughput, network lifetime, power consumption, packet delivery ratio) on the two WSNs with and without central node, we generated couple more simulations in the MRM environment using simulation settings in Table B in Cooja. The simulation results for the MRM environment are added in each of the sub-sections under sections 2.3.5 and 2.4.9 after the UDGM distance-loss model results.

SNR reception threshold (dB)	6
Background noise mean (dBm)	-100
Background noise variance (dB)	1
Extra system gain mean (dB)	0
Extra system gain variance (dB)	4
Frequency (MHz)	2400
Capture effect preamble (us)	64
Capture effect threshold (dB)	3
Default transmitter output power (dBm)	1.5
Directional antennas with TX gain	True
Receiver sensitivity (dBm)	-100
Directional antennas with RX gain	False
Use FSPL on total path lengths only	True
Max path rays	1
Max refractions	1
Max reflections	1
Max diffractions	0
Reflection coefficient (dB)	-3
Reflection coefficient (dB)	-5
Diffraction coefficient (dB)	-10
Obstacle attenuation (dB/m)	-3

Table B. MRM environment simulation settings

2.3.3 WSN setup in Cooja (without gateway device)

Aiming to create a simulation akin to the real world, we placed five low-power 2.4GHz, IEEE 802.15.4, and 6LowPAN compatible WSN Z1 motes from Zolertia [42] at random locations in Cooja, as in Figure 2. To implement decision-making in the case of WSN without a central node, we used a random number-based simple conditional logic to pick the destination node for sending a counter as the message.

Thus, to implement

```
if <condition>,
    Send <message> to <node>
```

we used

Condition: Pick a destination node if chosen by the random function

Message: Counter value

Node: Destination address chosen by the condition

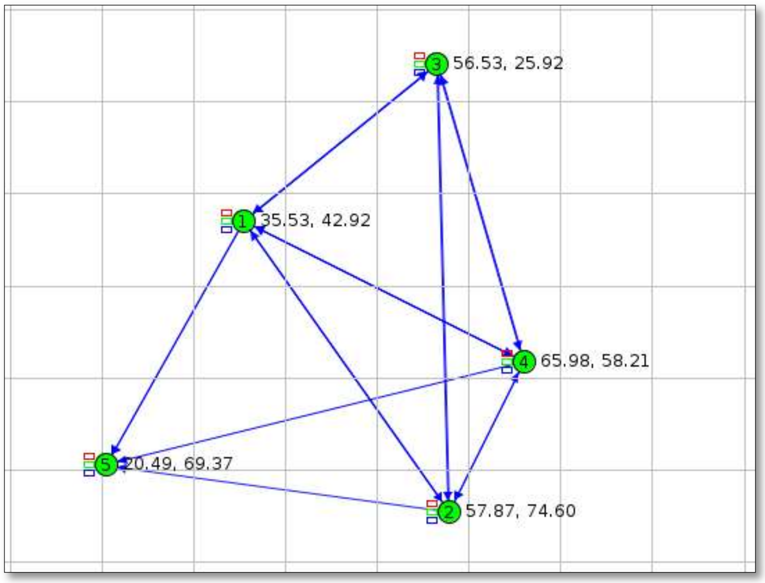


Figure 2(f). Z1 WSN Nodes at random locations in Cooja (with no central node) UDGM distance-loss model

2.3.4 Simulation Settings

The following settings are used while simulating a WSN without a gateway device using UDGM distance-loss model.

MAC Layer (PHY+MAC)	IEEE 802.15.4 CSMA
Number of nodes	5
Simulation Time	5 minutes 49 seconds
Energy Measurement Interval	Every second
Energy Measurement Module	Energest
Radio Propagation Model	Unit Disk Graph medium (UDGM) – Distance Loss
Node positioning	Random
Transmission (TX) Range	50 m
Communication Type	Peer to peer / Unicast
Network Topology	Random
Total Frame Size	25 bytes
MAC layer	CSMA
Net layer	Null net
Routing	Null Routing
Default Channel	26

Table 2. UDGM distance-loss model simulation settings.

Channel ID 26, shown as the default channel above, is one of the physical channels in a 2.4 GHz frequency range and has a center frequency of 2480.

2.3.5 Simulation Results

The below performance metrics are dependable indicators of Quality of Service (QoS) in wireless sensor networks [25] [54].

- Network Throughput
- Network Lifetime
- Packet Delivery Ratio
- Power consumption

The sub-sections under 2.3.5 and 2.4.8 evaluate the above QoS parameters for the two WSNs implemented with and without a central node, using UDGM distance-loss and MRM radio propagation models

2.3.5.1 Network Throughput

In data transmission, network throughput is the amount of data moved successfully from one place to another in a given time period and is typically measured in bits per second (bps). Higher throughput implies a better network [36] [53]. Tables 3(a) and 3(b) list the network throughput statistics from the Wireshark network analyzer tool [37] for the two radio propagation models.

Total Packets	22562
Total Simulation Time	349 seconds
Average packets per second	1.3
Average packet size	15 bytes
Average bytes per second	18
Average bits per second	151
Encapsulation	IEEE 802.15.4 Low-rate Wireless PAN

Table 3(a). Network throughput - UDGM distance-loss model

Throughput is sometimes measured as data packets per time slot [36]. Figures 3(a) and 3(b) show the input/output graph generated in the Wireshark for the throughput measured as packets per minute for the two radio propagation models.

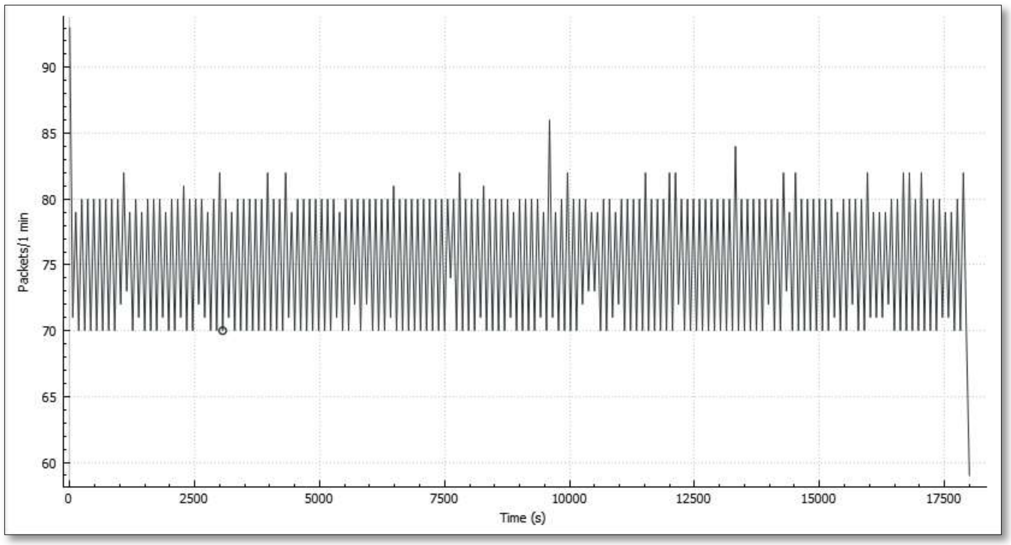


Figure 3(a). Packets per minute using CSMA-CA CC2420 RF Transceiver.
(For WSN without a central node) - UDGM distance-loss model

Total Packets	121296
Total Simulation Time	349 seconds
Average packets per second	5.7
Average packet size	22 bytes
Average bytes per second	127
Average bits per second	1021
Encapsulation	IEEE 802.15.4 Wireless PAN

Table 3(b). Network throughput in MRM environment

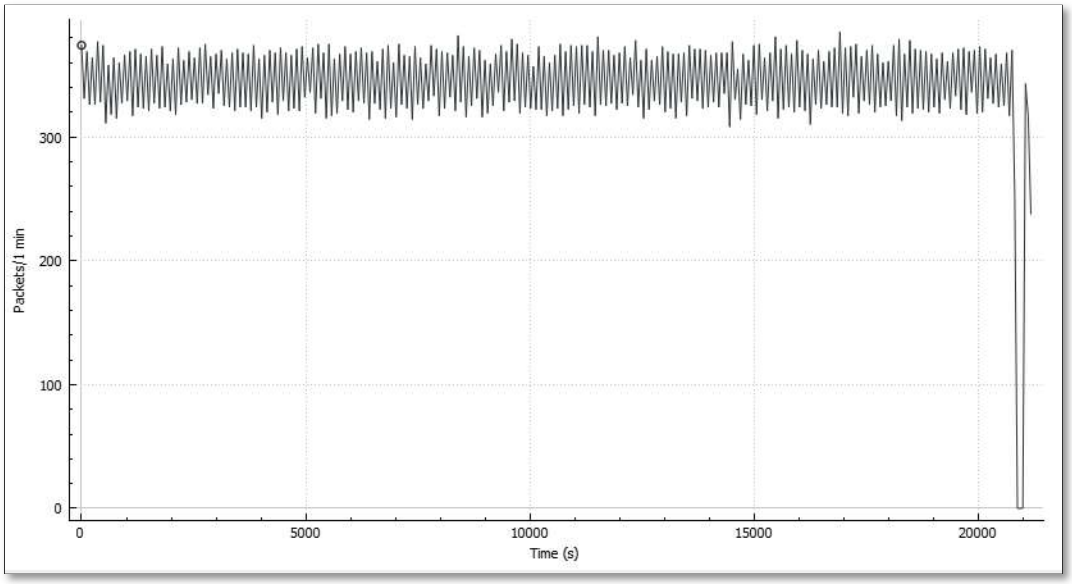


Figure 3(b). Packets per minute in MRM environment
(For WSN without a central node)

2.3.5.2 Network Lifetime

A lower RDC ratio indicates a larger network lifetime. A node must keep its radio off to conserve energy. Tables 4(a) and 4(b) provide the RDC ratio values of five motes, as recorded in Cooja for the two radio propagation models.

Mote	ON	TX	RX
Z1	100%	0.02%	0.05%
Z2	100%	0.02%	0.06%
Z3	100%	0.02%	0.05%
Z4	100%	0.02%	0.06%
Z5	100%	0.01%	0.07%

Table 4(a). Radio duty cycle percentage (For WSN without a central node) - UDGM distance-loss model

Mote	ON	TX	RX
Z1	99.96%	0.10%	0.18%
Z2	99.96%	0.10%	0.22%
Z3	99.95%	0.10%	0.21%
Z4	99.96%	0.11%	0.33%
Z5	99.95%	0.10%	0.24%

Table 4(b). Radio duty cycle percentage in MRM environment
(For WSN without a central node)

2.3.5.3 Power Consumption

The power consumption value (in mW unit) of a node in Contiki/Cooja [45] can be calculated using Equation [A]. Therefore, we will be required to find out various input values for this equation.

$$Power\ Consumption = \frac{Energest_{value} \times Current \times Voltage}{RTIMER\_SECOND \times Runtime}$$

Equation [A]

Under Contiki-NG, the Energest module keeps track of energy consumption by the CPU and Radio components of the sensors. Knowing how long the components have been in different states like active, radioactive, or stand-by, and the energy consumption in those states makes it possible to estimate the power consumption by the sensor nodes [29]. The average Energest values of the five nodes derived from the recorded Cooja simulation log for the two radio propagation models are given in Table 5(a) and 5(b).

Average Energest Value	Node1	Node2	Node3	Node4	Node5
CPU	6804.847	8116.051	6788.033898	6869.423729	5615.542373
LPM	1959274	1957963	1959290.78	1959209.39	1960463.271
TX	198	198.6441	195.4067797	201.2033898	195.3559322
RX	1965866	1965866	1965869.288	1965863.085	1965869.237

Table 5(a). Average Energest values (For WSN without a central node)
UDGM distance-loss model

Average Energest Value	Node1	Node2	Node3	Node4	Node5
CPU	10185.186	11559.948	10083.052	11818.690	10186.983
LPM	1955893	1954554	1955990	1954298	1955886
TX	1611.31	1611.41	1611.56	1611.92	1611.39
RX	1964403.271	1964403.458	1964408.224	1964402.576	1964402.881

Table 5(b). Average Energest values (For WSN without a central node) in MRM environment

Next, we work out the voltage and current consumption values of CPU and radio hardware components of the Z1 mote for use in Equation [A]. We begin with reproducing a few details of the Z1 WSN motes from the manufacturer data specification sheet [43] in Table 6.

WSN mote			Z1 Zolertia
Operating Voltage			3V
Microcontroller (CPU) unit	Type		MSP430 series
	Frequency		16 MHz
	Current Consumption	Active mode	< 10 mA
		Stand-by/ low-power mode (LPM)	0.5 μA
Radio Transceiver	Type		CC2420 Radio Frequency Transceiver
	Current Consumption	Transmit/TX mode	17.4 mA
		Receive/RX mode	18.8 mA

Table 6. Current, voltage and frequency values from the manufacturer data specification sheet [43].

The active mode CPU frequency of an emulated Z1 mote is 8 MHz by default in Cooja [44] and differs from its 16 MHz value specified in the manufacturer datasheet. We, therefore, need to investigate the current consumption of the emulated Z1 motes for their 8 MHz frequency in Cooja. Referring to [42], we adopt the methodology and the resultant active mode CPU current consumption value 4.3mA at 8 MHz for Z1 motes in Cooja.

Hence, specific to Cooja, the Z1 values we shall be using are:

CPU Frequency	8 MHz
Active mode CPU current consumption @ 8MHz	4.3 mA

Table 7. Adjusted active mode CPU current consumption at 8MHz in Cooja. Adopted from [42].

The energy consumption data is generated per unit time of the hardware clock known as RTIMER_SECOND. And the RTIMER_SECOND value for Z1 mote is 32768 ticks per second in Cooja [45]. The length of time for which the simulation is run is called the *Run time*, and it is the difference of simulation start and end time which is 349 seconds in our case.

Thus, for Z1 motes, we can summarize the standard operating voltage, current consumption values for CPU and Radio Transceiver components, RTIMER_SECOND, and simulation run time in the table below.

Z1 mote parameter	**Value**
Normal operating Voltage	3V
CPU Active state current consumption @ 8 MHz	4.3 mA
CPU Standby/LPM current consumption	0.5 μA
TX CC2420 RF Transceiver current consumption	17.4 mA
RX CC2420 RF Transceiver current consumption	18.8 mA
RTIMER_SECOND (real timer)	32768 ticks per second
Total simulation Runtime	349 seconds

Table 8. Values for Z1 mote for use in power calculations.

We use the values from Table 8 and apply them in Equation [A] to calculate the power consumption values for the five nodes in the two radio propagation models, as shown in Tables 9(a) and 9(b)

Power(mW)	**Node1**	**Node2**	**Node3**	**Node4**	**Node5**
CPU	0.0077	0.0092	0.0077	0.0077	0.0063
LPM	0.0003	0.0003	0.0003	0.0003	0.0003
TX	0.0009	0.0009	0.0009	0.0009	0.0009
RX	9.6952	9.6952	9.6952	9.6952	9.6952
Total	**9.7041**	**9.7056**	**9.7041**	**9.7041**	**9.7027**

Table 9(a). Power consumption (For WSN without a central node)- UDGM distance-loss model

Power(mW)	**Node1**	**Node2**	**Node3**	**Node4**	**Node5**
CPU	0.0115	0.0130	0.0114	0.0133	0.0115
LPM	0.0003	0.0003	0.0003	0.0003	0.0003
TX	0.0074	0.0074	0.0074	0.0074	0.0074
RX	9.6880	9.6880	9.6880	9.6880	9.6880
Total	**9.7072**	**9.7087**	**9.7070**	**9.7089**	**9.7071**

Table 9(b). Power consumption (For WSN without a central node) in MRM environment

2.3.5.4 Packet Delivery Ratio (PDR)

A network's performance is measured by the packet delivery ratio (PDR) given by the equation below [25].

$$PDR(\%) = \frac{(Number\ of\ the\ received\ packets)}{(Number\ of\ the\ transmitted\ packets)} \times\ 100$$

Equation [B]

Tables 10(a) and 10(b) show the number of packets each node sent and received in our Cooja simulations for the two radio propagation models.

Source Node	Packet Sent	Packet Received	PDR
1	449	449	100%
2	449	449	100%
3	449	449	100%
4	449	448	99.8%
5	449	449	100%

Table 10(a). Packet delivery ratio (For WSN without a central node) UDGM distance-loss model

Source Node	Packet Sent	Packet Received	PDR
1	21024	21024	100%
2	21024	42048	200%
3	21024	21035	100.05%
4	21035	21024	99.95%
5	21024	0	0%

Table 10(b). Packet delivery ratio (For WSN without a central node) in MRM environment

The PDR for node 2 in the MRM environment is double due to its proximity with nodes 3 and 4, as shown in Figure 3(c).

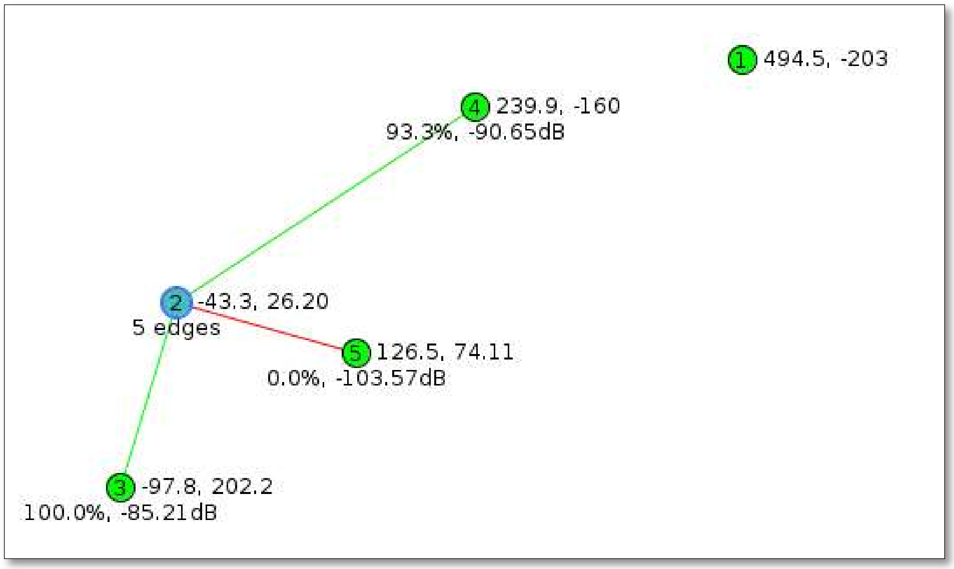
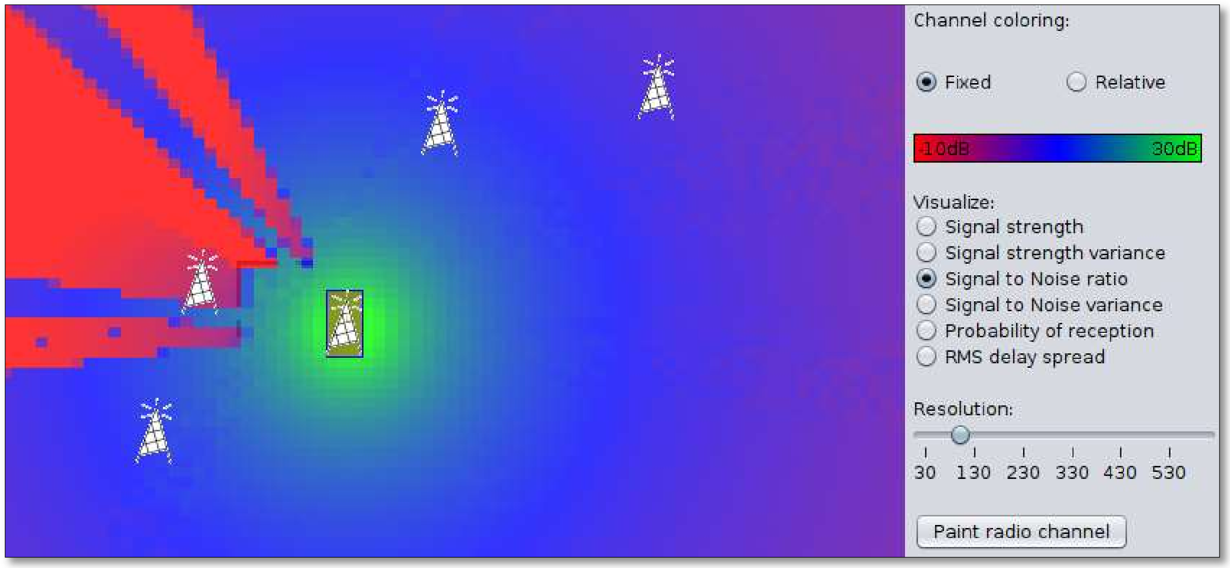


Figure 3(c). Reachability of node 2 in MRM environment

To investigate the reason behind the 0% PDR of node 5, we refer to the SNR reception threshold of 6 dB in the MRM settings in Table B. Thus a node cannot receive signals with SNR below 6 dB in the given MRM environment. We observed the signal-to-noise ratio around node 5 in the color-painted radio channel in Figure 3(d) and saw that the obstacle and the large distances impacted the ability of node 5 to receive packets from the other nodes.



Channel coloring: [✓ Fixed] Relative

Visualize: [✓ Signal to Noise ratio]

Figure 3(d). signal to noise ratio in colored radio channel around node surrounded with obstacle

2.4 Simulation of a distributed WSN with central/root node(s)

This section investigates RPL 6LowPAN wireless networks that work around a central node (also known as root or sink node). The RPL networks are the most successful for IPv6-based networking in low-powered and lossy network (LLN) devices and incorporate the latest improvements in wireless sensor networking [30]. These networks are built as Directed Acyclic Graphs (DAGs), where the paths are oriented toward and must terminate in a root node with no outgoing edges [51]. For multipath routing in distributed networks, the RPL networks use Destination-Oriented Directed Acyclic Graphs (DODAGs) terminating at a single root or border router. When acting as a router, the central node is referred to as the low-powered and lossy border router (LBR) device.

2.4.1 Importance of the central node

The root node is indispensable and central to the design of RPL networks. The central node is often deployed first and should be constantly powered. It is considered dependable enough to provide connectivity among non-root LLN devices. In essence, a minimal configuration non-root sensor with no in-built network stack can still communicate with the neighboring nodes or the upper network layers via the root node.

The central node as an LBR device enables router-based, distributed, wide-area, multi-sensor networks spread over geographies and extended to global networks via IPv6 connectivity. The communication interfaces for web or messaging can be extended beyond LBR using Message Queuing Telemetry Transport (MQTT) and Constrained Application (CoAP) protocols [42].

2.4.2 Neighbor Discovery Mechanism Using Central/Root Node

Described in section 2.3.1, the IPv6 network discovery in the low-power IPv6 stack under Contiki-NG is based on RFC 4861, where "*IPv6 nodes on the same link use Neighbor Discovery to discover each other's presence, to determine each other's link-layer addresses, to find routers, and to maintain reachability information about the paths to active neighbors*" [61][62].

However, for 6LowPAN RPL networks dependent on the root nodes under Contiki-NG, the "*Default RPL NBR policy decides when to add a new discovered node to the nbr table from RPL. This policy assumes that all neighbors end up being IPv6 neighbors and are not only MAC neighbors*" [63]. Thus, besides the IPv6 network discovery framework, the existence of the nodes in RPL networks is determined by a neighbor policy dependent on the RPL storing mode. With the storing mode-ON, all nodes cache the neighboring node addresses, whereas in the case of storing mode off, only the root node caches all the addresses. The caching of nodes impacts the discoverability of the nodes. Therefore, RPL uses different routing approaches described under sections 2.4.3, 2.4.5, and 2.4.6 for different storing modes.

2.4.3 Communication modes in RPL networks

For our use-case of distributed WSN, we essentially need direct communication capabilities among non-root sensors (or child nodes or clients). Since "*the edge node is mostly one or two hops away from the mobile client to meet the response time constraints for real-time*" [18, 19]. The edge-AI algorithms should execute near the central device and not necessarily on the central device. So, we explore the routing mechanisms available in RPL networks to establish autonomy and communication among the non-root nodes.

For passing control messages with node addresses, RPL uses two routing modes: storing and non-storing, as shown in Figure 4. In non-storing mode, the message is sent directly to the root, the only node that can maintain the addresses. However, a child node's discovery of neighboring nodes becomes possible when it begins recording addresses in a routing table in its memory under the storing mode [28]. The routing table contains, among other things, the identifier for each of the other sensors. A child node can thus communicate with any other node when the RPL network is implemented in the storing mode.

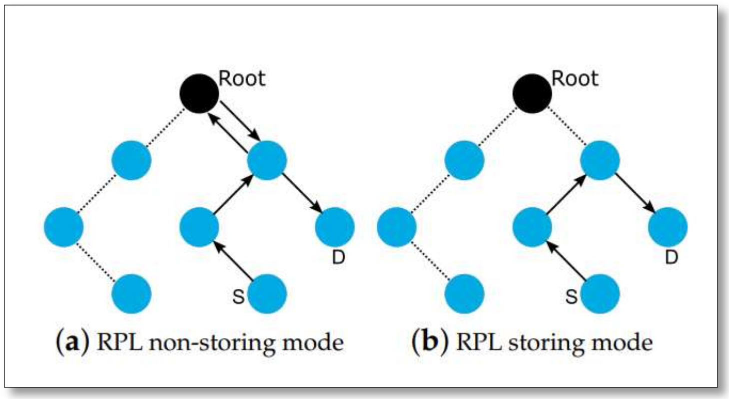


Figure 4. Routing modes in an RPL network. Adopted from [21].

Figure 4(a) shows the default non-storing mode and involves passing by root. Figure 4(b) shows RPL with storing mode ON, where the routing happens through the common ancestor between the source and destination nodes instead of the root.

2.4.4 RPL network with UDP communication

We extend the “rpl-udp” example [52] from the Contiki-NG repository in this work. The example is a simple RPL network with User Datagram Protocol (UDP) communication, containing a DAG root and client. The clients periodically send a UDP request containing a counter as a payload. The server responds with the same counter back to the originator upon receiving the request.

We implement autonomous decision-making in the RPL-UDP root and client nodes. And enhance the child nodes to be able to talk to their peers directly via building and maintaining an independent routing table.

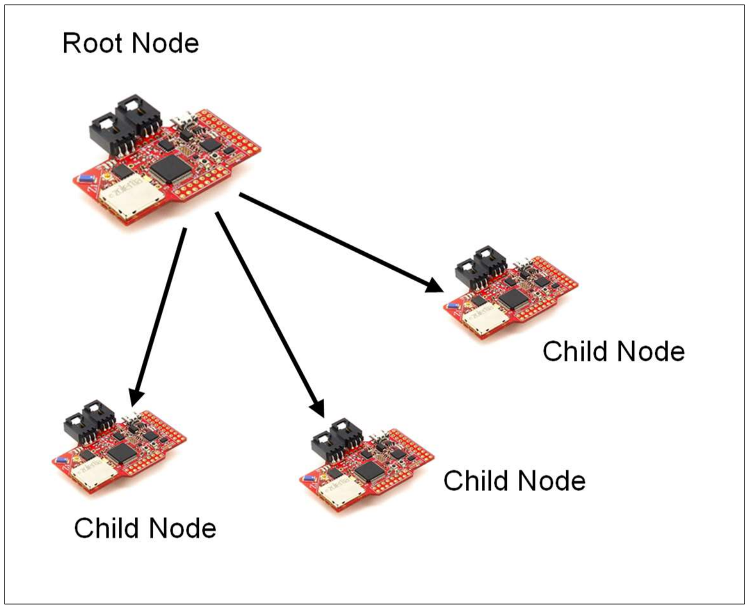
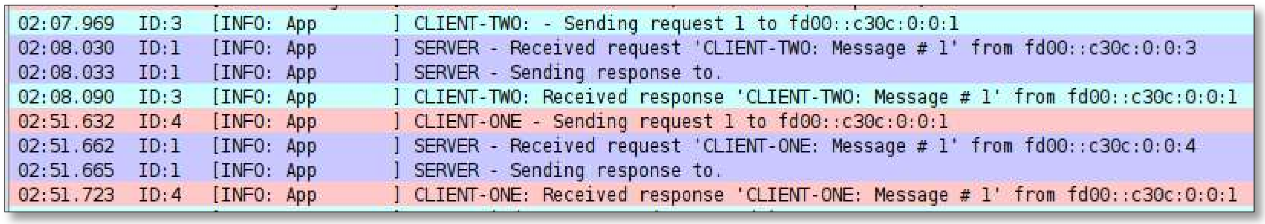


Figure 5. Z1 WSN with a central node.

2.4.5 Root Node in RPL-UDP – using default non-storing mode

Using the default non-storing mode of RPL networks, we verified via a few simulation tests that the clients can communicate (send/receive messages) only via the server node and not directly with each other. For example, In Figure 6 from the Cooja simulation log, we see that the client node id 3 (CLIENT-TWO) and node id 4 (CLIENT-ONE) can send the request to server node 1 and can receive the same message back from node 1. But the communication never directly happens between the client node ids 3 and 4.



```
02:07.969   ID:3   [INFO: App        ] CLIENT-TWO: - Sending request 1 to fd00::c30c:0:0:1
02:08.030   ID:1   [INFO: App        ] SERVER - Received request 'CLIENT-TWO: Message # 1' from fd00::c30c:0:0:3
02:08.033   ID:1   [INFO: App        ] SERVER - Sending response to.
02:08.090   ID:3   [INFO: App        ] CLIENT-TWO: Received response 'CLIENT-TWO: Message # 1' from fd00::c30c:0:0:1
02:51.632   ID:4   [INFO: App        ] CLIENT-ONE - Sending request 1 to fd00::c30c:0:0:1
02:51.662   ID:1   [INFO: App        ] SERVER - Received request 'CLIENT-ONE: Message # 1' from fd00::c30c:0:0:4
02:51.665   ID:1   [INFO: App        ] SERVER - Sending response to.
02:51.723   ID:4   [INFO: App        ] CLIENT-ONE: Received response 'CLIENT-ONE: Message # 1' from fd00::c30c:0:0:1
```

Figure 6. Cooja message log

2.4.6 RPL – Storing Mode in Contiki

As discussed in section 2.4.3, with the storing mode ON under Contiki RPL networks, each client builds its routing table for accessing the neighboring nodes. We can set the storing mode on via the following setting in the Cooja project Makefile [28]:

MAKE_ROUTING = MAKE_ROUTING_RPL_CLASSIC

Figure 7 shows an enhanced 6LowPAN RPL-UDP network with node 1 (green circle) as the root node and the remaining as client nodes (yellow circles). Their IPv6 addresses are marked ending with node id numbers, e.g., IPv6 address for node 1 is **fe80::c30c:0:0:1** and so on.

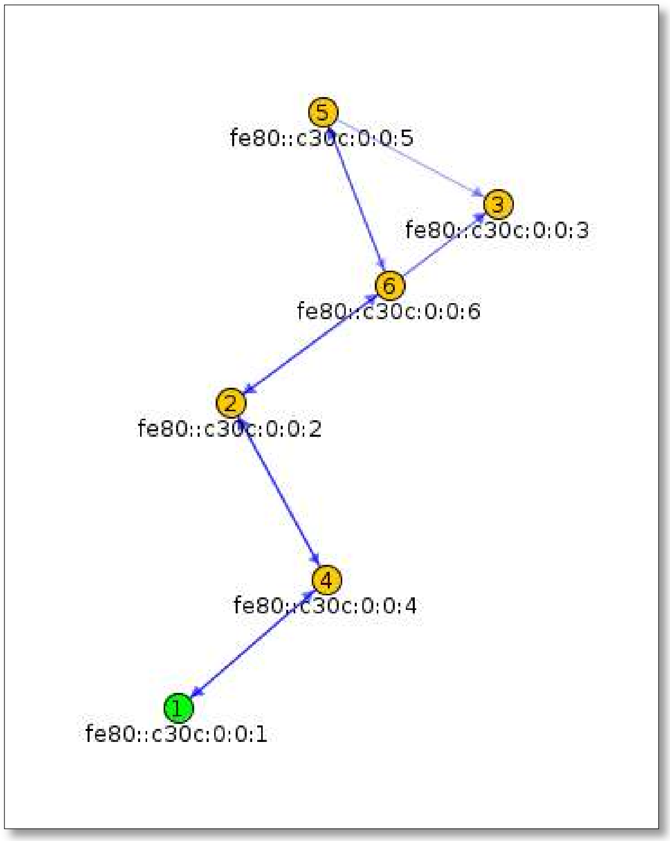
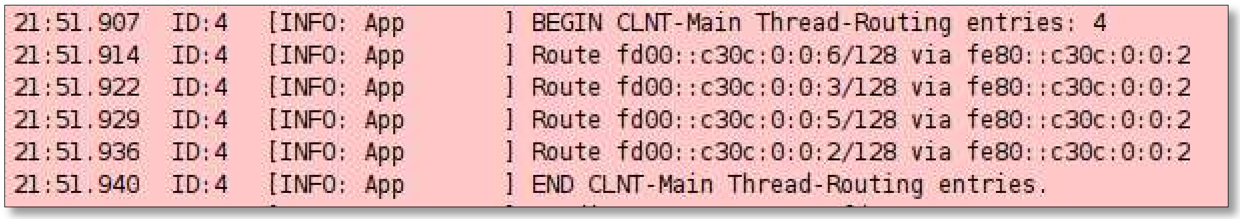


Figure 7. Tree graph formed in UDP-RPL.

Figure 8 shows the routing table built by node 4.



```
21:51.907  ID:4  [INFO: App      ] BEGIN CLNT-Main Thread-Routing entries: 4
21:51.914  ID:4  [INFO: App      ] Route fd00::c30c:0:0:6/128 via fe80::c30c:0:0:2
21:51.922  ID:4  [INFO: App      ] Route fd00::c30c:0:0:3/128 via fe80::c30c:0:0:2
21:51.929  ID:4  [INFO: App      ] Route fd00::c30c:0:0:5/128 via fe80::c30c:0:0:2
21:51.936  ID:4  [INFO: App      ] Route fd00::c30c:0:0:2/128 via fe80::c30c:0:0:2
21:51.940  ID:4  [INFO: App      ] END CLNT-Main Thread-Routing entries.
```

Figure 8. Routing Table automatically built at the client-node

Thus, from Figures 7 and 8, we can conclude for node 4 that it has the below paths available to access the neighboring nodes 3, 5, and 6

- Node 4 can access node 6 via node 2.
- Node 4 can access node 3 via node 2.
- Node 4 can access node 5 via node 2.
- Node 4 can access node 2 via node 2 itself.

This access-graph/routing table changes if the node positions are altered, that is, when the nodes move or disappear, e.g., due to a change in their relative position or due to a malfunction, etc. The routing table is built afresh when a trigger arrives at a sensor, which makes a sensor detect changes in the state of its neighboring sensor.

Thus, with storing mode-on in Contiki, we observe that:

1. Each sensor can discover other sensors in its neighborhood without the help of any central server (storage, processor) or root node.
2. The network is dynamic, i.e., when sensors enter or exit the network, the nodes can detect the live positions/addresses of other nodes using the dynamic routing table.

2.4.7 Point to Point (P2P) RPL protocol

The Point-to-point RPL (P2P RPL) [20] is an extension of RPL that implements shorter paths that do not pass by the sink (or root node.) It uses an IPv6 router instead of an RPL router on 6LoWPAN networks and provides the capability to link arbitrary paths connecting point-to-point nodes.

The P2P-RPL came as RFC 6997 [20] in August 2013, after the RPL RFC 6550 [22] that came in March 2012. Since many shortcomings of even P2P-RPL are noted, newer improved P2P variations like GeoRank, ER-RPL, AODV-RPL [21] and alternatives like the Lightweight On-demand Ad hoc Distance-vector Routing Protocol – Next Generation (LOADng) [29, 30, 31, 32], Babel Routing Protocol RFC 8965 [24] have emerged with better routing for LLN devices.

2.4.8 WSN setup in Cooja (with root node)

Figures 8(a) and 8(b) show two 6LowPAN RPL-UDP WSNs with one server node surrounded by four client nodes. We used these WSNs to generate simulations using the UDGM distance-loss and the MRM radio propagation models. The MRM simulation included an obstacle (Figure 8(c)).

To implement decision-making in the case of WSNs with root nodes, we send the counter message to a destination node if that node is present in the dynamic routing table of the source node.

Thus, to implement

```
        if <condition>,
                Send <message> to <node>
```

we used

Condition: Pick destination node if present in dynamic routing table of the source node

Message: Counter value

Node: Destination address chosen by the condition

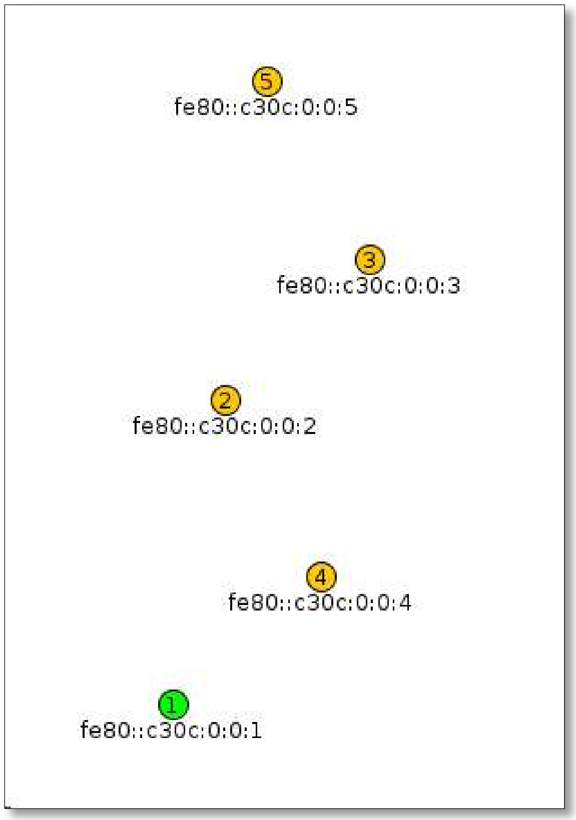


Figure 8(a). RPL WSN in Cooja with a central node- UDGM distance-loss

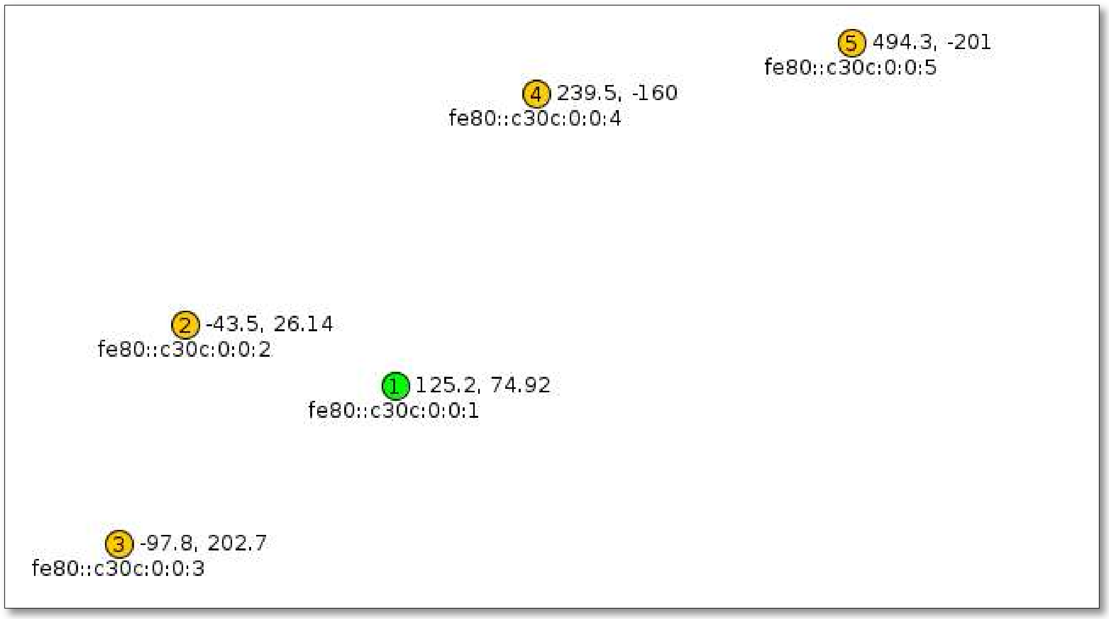


Figure 8(b). RPL WSN in Cooja with a central node in MRM environment

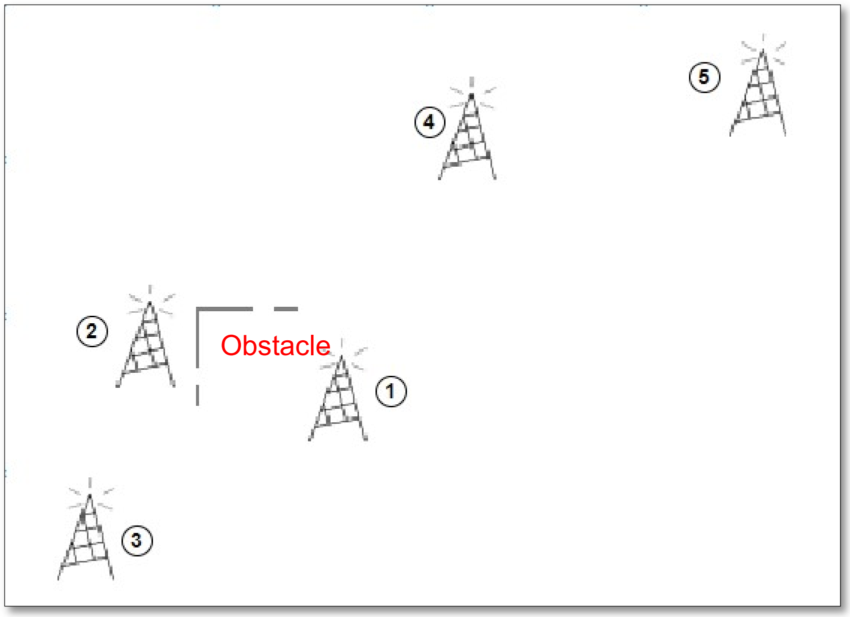


Figure 8(c). An obstacle placed around root node 1 in the MRM environment

2.4.9 Simulation Results

2.4.9.1 Network Throughput

Total Packets	22824
Total Simulation Time	349 seconds
Average packets per second	1.1
Average packet size	35 bytes
Average bytes per second	38
Average bits per second	307
Encapsulation	IEEE 802.15.4 Wireless PAN

Table 11(a). Network throughput (For WSN with a central node)- UDGM distance-loss model

Total Packets	12910
Total Simulation Time	349 seconds
Average packets per second	0.6
Average packet size	38 bytes
Average bytes per second	23
Average bits per second	189
Encapsulation	IEEE 802.15.4 Wireless PAN

Table 11(b). Network throughput (For WSN with a central node) in MRM environment

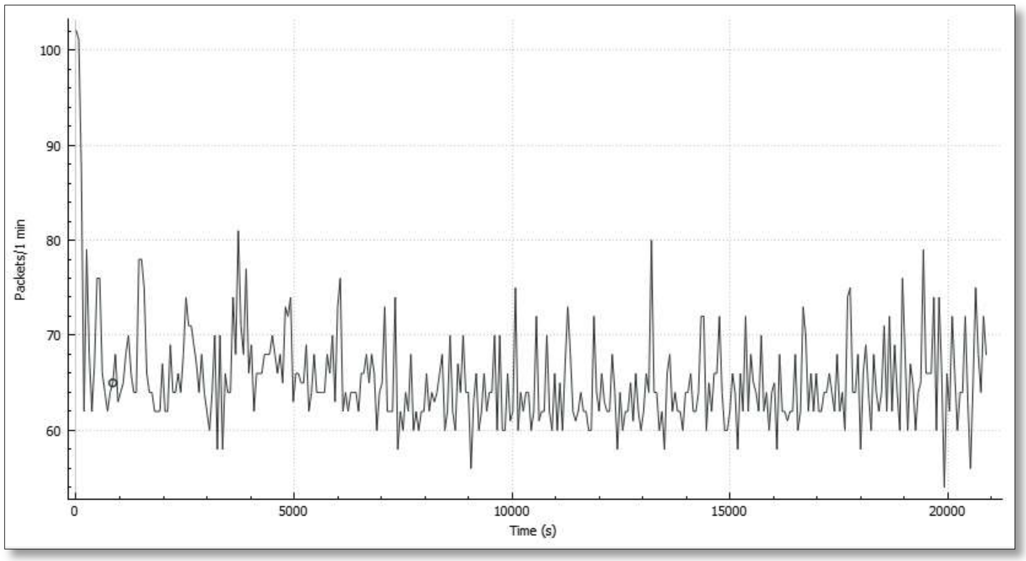


Figure 9(a). Packets per minute (For WSN with a central node)- UDGM distance-loss model

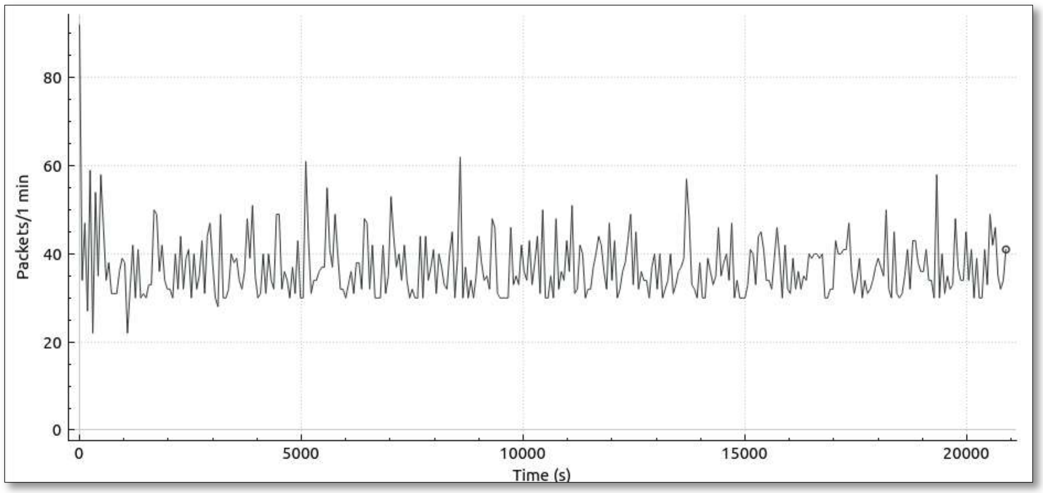


Figure 9(b). Packets per minute (For WSN with a central node) in MRM environment

2.4.9.2 Network Lifetime

Mote	ON	TX	RX
Z1 (root)	99.97%	0.017%	0.043%
Z2	99.98%	0.037%	0.071%
Z3	99.93%	0.028%	0.052%
Z4	99.86%	0.043%	0.054%
Z5	99.99%	0.015%	0.028%

Table 12(a). Radio duty cycle percentage (For WSN with a central node)
UDGM distance-loss model

Mote	ON	TX	RX
Z1 (root)	99.98%	0.02%	0.06%
Z2	99.81%	0.01%	0.06%
Z3	99.88%	0.01%	0.04%
Z4	99.81%	0.02%	0.05%
Z5	99.87%	0.02%	0.04%

Table 12(b). Radio duty cycle percentage (For WSN with a central node) in MRM environment

2.4.9.3 Power Consumption

Average Energest Value	Node1	Node2	Node3	Node4	Node5
CPU	11101.778	42394.49153	41163.424	43306.32203	38933.169
LPM	1955164.119	1957006.424	1958236.271	1956094.61	1960466.525
TX	33569.508	33948.18644	33795.152	34027.55932	33583.288
RX	1965824.153	1965443.305	1965596.881	1965362.559	1965811.068

Table 13(a). Average Energest values (For WSN with a central node)
UDGM distance-loss model

Average Energest Value	Node1	Node2	Node3	Node4	Node5
CPU	6799.53	5957.80	5415.98	6956.92	5559.05
LPM	1959279.288	1960121.475	1960663.153	1959122.356	1960519.78
TX	254.12	224.90	241.41	386.05	343
RX	1965819.576	1965851.237	1965833.707	1965687.068	1965731.525

Table 13(b). Average Energest values (For WSN with a central node) in MRM environment

We use the values from Tables 13(a) and 13(b) and apply them in Equation [A] to calculate the power consumption values for the five nodes in the WSN with a central node in Table 14(a) and 14(b) for the two radio propagation models.

Power(mW)	**Node1**	**Node2**	**Node3**	**Node4**	**Node5**
CPU	0.0125	0.0478	0.0464	0.0489	0.0439
LPM	0.0003	0.0003	0.0003	0.0003	0.0003
TX	0.1532	0.1550	0.1543	0.1553	0.1533
RX	9.6950	9.6931	9.6939	9.6927	9.6949
Total	**9.8610**	**9.8962**	**9.8948**	**9.8972**	**9.8924**

Table 14(a). Power consumption (For WSN with a central node)- UDGM distance-loss model

Power(mW)	**Node1**	**Node2**	**Node3**	**Node4**	**Node5**
CPU	0.0077	0.0067	0.0061	0.0078	0.0063
LPM	0.0003	0.0003	0.0003	0.0003	0.0003
TX	0.0012	0.0010	0.0011	0.0018	0.0016
RX	9.6950	9.6951	9.6951	9.6943	9.6946
Total	**9.7041**	**9.7032**	**9.7026**	**9.7042**	**9.7027**

Table 14(b). Power consumption (For WSN with a central node) in MRM environment

2.4.9.4 Packet Delivery Ratio (PDR)

Source Node	Node IPv6 address	Packet Sent (TX)	Packet Received	PDR
1 (Root)	fe80::c30c:0:0:1	13	421	3238.5%
2	fe80::c30c:0:0:2	446	368	82.5%
3	fe80::c30c:0:0:3	416	461	110.8%
4	fe80::c30c:0:0:4	480	350	72.9%
5	fe80::c30c:0:0:5	412	80	19.4%

Table 15(a). Packet Delivery Data (For WSN with a central node)- UDGM distance-loss model

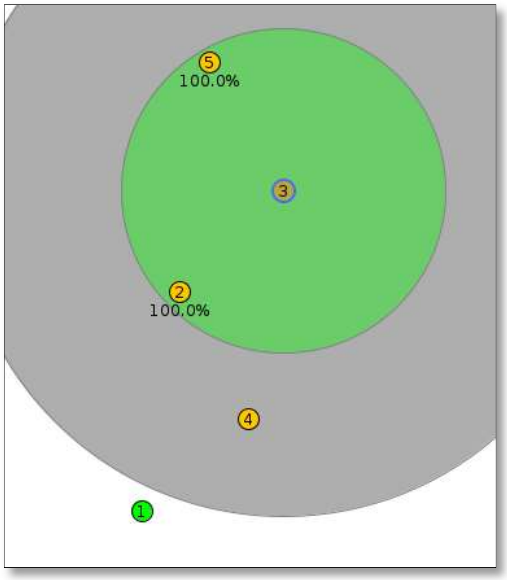


Figure 10. Transmission range of node 3 in RPL network

Node 3 had more packets received than sent due to nodes 5 and 2 being in its access radius.

Source Node	Node IPv6 address	Packet Sent (TX)	Packet Received	PDR
1 (Root)	fe80::c30c:0:0:1	31	112	361.3%
2	fe80::c30c:0:0:2	330	337	102.1%
3	fe80::c30c:0:0:3	423	405	95.7%
4	fe80::c30c:0:0:4	207	187	90.3%
5	fe80::c30c:0:0:5	569	376	66.1%

Table 15(b). Packet Delivery Data (For WSN with a central node) in MRM environment

Table 15 (b) shows that node 5 has the lowest PDR due to its long distance from other nodes under the MRM environment. But the PDR at node 5 is not 0 due to the access path tree formed in the RPL network.

3 Conclusions

I. We simulated the distributed wireless sensor networks using Contiki-NG / Cooja where the nodes communicate directly with the other nodes -
   - without having to go through a central node when implemented via non-IPv6 lower layers
   - via the IPv6 under RPL 6LoWPAN networks without having to go through the root node when running Contiki under storing mode

II. Using the mechanisms described on Non-IPv6 MAC/Nullnet layer networks in section 2.3 and for RPL networks in section 2.4.2, we built two WSN simulations that demonstrate
   - asynchronous event-triggered communication - because the nodes send packets based on conditional logic implemented in the source code of the emulated nodes in ContikiOS/Cooja
   - targeted communication - because the nodes send packets to the specific recipient nodes using ContikiMAC and RPL protocols from the network stack of the ContikiOS
   - dynamic communication - because the nodes detect the target nodes using ContikiMAC carrier sensing and RPL routing mechanism

III. Comparing the simulation results for the WSNs using UDGM distance-loss model with and without central node, we observe that
   - **Network Throughput**: From tables 3(a) and 11(a), it is observed that RPL WSN with a central node has lesser network throughput
   - **Network Lifetime**: From tables 4(a) and 12(a), it is observed that RPL WSN with a central node has a lower network lifetime

- **Power Consumption**: From tables 9(a) and 14(a), it is observed that RPL WSN with a central node has a higher power consumption
- **Packet Delivery Ratio**: From tables 10(a) and 15(a), it is observed that RPL WSN with a central node has a degraded PDR for the four child nodes.

IV. Comparing the simulation results for the WSNs in MRM environment with and without central node, we observe that

- **Network Throughput**: From tables 3(b) and 11(b), it is observed that WSN without the central node has significantly higher network throughput
- **Network Lifetime**: From tables 4(b) and 12(b), it is observed that RPL WSN with a central node has a higher network lifetime
- **Power Consumption**: From tables 9(b) and 14(b), it is observed that WSN without the central node has slightly higher total power consumption due to higher CPU and Transmission power consumption
- **Packet Delivery Ratio**: From tables 10(b) and 15(b), it is observed that obstacles and larger distances among the nodes negatively impact the packet delivery ratio in both types of WSNs

Thus, we see that, on the one hand, the edge devices can yield actionable real-time insights, processing the AI or machine learning algorithms. On the other hand, the WSN networks can deliver continuous integration beyond the edge of the network, addressing constraints of low-powered and lossy networks.

4 Appendix

Table of Acronyms

WSN	Wireless Sensor Network
Edge-AI	Edge Artificial Intelligence
RPL	Routing Protocol for Low-Power and Lossy Networks
QoS	Quality of Service
IEEE	Institute of Electrical and Electronics Engineers
AI	Artificial Intelligence
ML	Machine Learning
IoT	Internet of Things
OMNeT++	Objective Modular Network Testbed in C++
NS-3	Network Simulator version 3
MATLAB	MATrix LABoratory
Contiki-NG	Contiki-Next Generation
CPU	Central Processing Unit
GUI	Graphical User Interface
GNU GPL	GNU's Not UNIX General Public License
ROM	Read only memory
RAM	Random Access Memory
TCP/IP	Transmission Control Protocol/Internet Protocol
BAN	Body Area Networks
DCE	Direct-Code Execution
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks
CoAP	Constrained Application Protocol
TSCH	Time Slotted Channel Hopping or Time Synchronized Channel Hopping
OSPF	Open Shortest Path First
BGP	Border Gateway Protocol
PPP	Point-to-Point Protocol
MAC	Medium Access Control
MANET	Mobile Ad Hoc Network
DiffServ	Differentiated services
MPLS	Multi-Protocol Label Switching
LDP	Label Distribution Protocol
RSVP-TE	Resource Reservation Protocol for Traffic Engineering
UDP	User Datagram Protocol
NSC	Network Simulation Cradle
BSD	Berkeley Software Distribution
XCP	Universal Measurement and Calibration Protocol
RS 232	Recommended Standard 232
UDGM	Unit Disk Graph Medium
DGRM	Directed Graph Radio Medium
MRM	Multipath Ray-tracer Medium
SBR	Shooting and Bouncing Rays
WMN	Wireless Mesh Network

OSI	Open System Interconnections
LR-WPAN	Low-Rate Wireless Personal Area Network
RDC	Radio Duty Cycle
LPL	Low-power Probing and low-power Listening
PDR	Packet Delivery Ratio
EUI	Extended Unique Identifier
Wi-Fi	Wireless Fidelity
TX	Transmission
RX	Reception
PHY	Physical
CSMA	Carrier Sense Multiple Access
GHz	Giga Hertz
mW	Milli Watt
RTIMER	Real Timer
LPM	Low Power Mode
RF	Radio Frequency
DAG	Directed Acyclic Graph
DODAG	Destination-Oriented Directed Acyclic Graph
LLN device	Low-powered and lossy network device
LBR	Low-powered and lossy Border Router
MQTT	Message Queuing Telemetry Transport
CoAP	Constrained Application Protocol
P2P	Point to Point
RFC	Request for Comments
LOADng	Lightweight On-demand Ad hoc Distance-vector Routing Protocol – Next Generation
ER-RPL	Energy-efficient region-based Routing Protocol
AODV	Ad-hoc On-demand Distance Vector
MRM	Multi-path Ray-tracer Medium
SNR	Signal to Noise Ratio

5 References

1. “Message Passing.” Wikipedia, Wikimedia Foundation, 21 Apr. 2022, https://en.wikipedia.org/wiki/Message_passing.
2. Kanaris, Loizos, et al. “On the Realistic Radio and Network Planning of IOT Sensor Networks.” MDPI, Multidisciplinary Digital Publishing Institute, 24 July 2019, https://www.mdpi.com/1424-8220/19/15/3264.
3. “RPL UDP.” RPL UDP - Contiki, https://anrg.usc.edu/contiki/index.php/RPL_UDP.
4. “Network Stack.” Network Stack - Contiki, https://anrg.usc.edu/contiki/index.php/Network_Stack.
5. Ali, Qutaiba I. “Chapter: Simulation Framework of Wireless Sensor Network (WSN) Using MATLAB/Simulink Software.” IntechOpen, IntechOpen, 26 Sept. 2012, https://www.intechopen.com/chapters/39337.
6. Introduction to Network Simulator Version 3 (NS-3). http://eng.staff.alexu.edu.eg/~bmokhtar/courses/computer_networks_ssp/fall_2014/ns3.pdf.
7. “IOSR Journals.” IOSR Journal, https://www.iosrjournals.org/iosr-jece/papers/Vol.%2015%20Issue%205/Series-1/A1505010106.pdf.
8. Unit II Data-Link Layer & Media Access - SNS Courseware. http://www.snscourseware.org/snscenew/files/1565835024.pdf.
9. “Contiki Programming Guide.” Contiki Programming Guide - Contiki, https://anrg.usc.edu/contiki/index.php/Contiki_Programming_Guide.
10. “ Castalia.” Castalia, https://omnetpp.org/download-items/Castalia.html.
11. “NS3 Tutorial.” Ns, https://www.nsnam.org/docs/release/3.35/tutorial/html/index.html.
12. “Contiki.” Wikipedia, Wikimedia Foundation, 28 Apr. 2022, https://en.wikipedia.org/wiki/Contiki.
13. “Events and Simulator.” Events and Simulator - Manual, https://www.nsnam.org/docs/release/3.29/manual/html/events.html.
14. “Introduction.” Introduction - INET 4.3.0 Documentation, https://inet.omnetpp.org/docs/users-guide/ch-introduction.html.
15. “Realtime.” RealTime - Manual, https://www.nsnam.org/docs/release/3.29/manual/html/realtime.html.
16. Contiki-Ng. “Releases · Contiki-Ng/Contiki-Ng.” GitHub, https://github.com/contiki-ng/contiki-ng/releases.
17. “Routing Table.” Wikipedia, Wikimedia Foundation, 4 Apr. 2022, https://en.wikipedia.org/wiki/Routing_table.
18. Tirado, Juan M. “We Have to Define What Is Edge Computing.” Jmtirado.net, 23 Sept. 2020, https://jmtirado.net/define_edge_computing/.
19. “Edge Computing.” Wikipedia, Wikimedia Foundation, 3 May 2022, https://en.wikipedia.org/wiki/Edge_computing.
20. “RFC 6997 - Reactive Discovery of Point-to-Point Routes in Low-Power and Lossy Networks.” Document Search and Retrieval Page, https://datatracker.ietf.org/doc/html/rfc6997.
21. Sobral, José V. V., et al. “Routing Protocols for Low Power and Lossy Networks in Internet of Things Applications.” MDPI, Multidisciplinary Digital Publishing Institute, 9 May 2019, https://www.mdpi.com/1424-8220/19/9/2144.
22. “RFC 6550 - RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks.” Document Search and Retrieval Page, https://datatracker.ietf.org/doc/html/rfc6550.
23. ArXiv. https://arxiv.org/ftp/arxiv/papers/1902/1902.01876.pdf.
24. RFC Editor, https://www.rfc-editor.org/rfc/rfc8965.txt.

25. QoS Measurement of RPL Using Cooja Simulator and Wireshark Network Analyser. https://www.researchgate.net/publication/325593751_QoS_Measurement_of_RPL_using_Cooja_Simulator_and_Wireshark_Network_Analyser.
26. Researching and Hardware Implementation of RPL Routing Protocol ... - IJFCC. http://www.ijfcc.org/papers/338-F1004.pdf.
27. “Scholars Crossing: Liberty University Research.” Site, https://digitalcommons.liberty.edu/cgi/viewcontent.cgi?article=2048&context=honors.
28. Contiki-Ng. “Documentation: RPL · Contiki-Ng/Contiki-Ng Wiki.” GitHub, https://github.com/contiki-ng/contiki-ng/wiki/Documentation:-RPL#modes-of-operation-mop.
29. Contiki-Ng. “Documentation: Energest · Contiki-Ng/Contiki-Ng Wiki.” GitHub, https://github.com/contiki-ng/contiki-ng/wiki/Documentation:-Energest.
30. Musaddiq, A., Zikria, Y.B., Zulqarnain et al. Routing protocol for Low-Power and Lossy Networks for heterogeneous traffic network. J Wireless Com Network 2020, 21 (2020). https://doi.org/10.1186/s13638-020-1645-4
31. Zarrad A, Alsmadi I. Evaluating network test scenarios for network simulators systems. International Journal of Distributed Sensor Networks. October 2017. doi:10.1177/1550147717738216
32. Obermaier, C.; Facchi, C. Observations on OMNeT++ Real-Time Behaviour. arXiv 2017, arXiv:1709.02207.
33. Varga, Andras. “Chapters.” OMNeT++ - Simulation Manual, https://doc.omnetpp.org/omnetpp/manual/.
34. Qutaiba I. Ali (September 26th 2012). Simulation Framework of Wireless Sensor Network (WSN) Using MATLAB/SIMULINK Software, MATLAB - A Fundamental Tool for Scientific Computing and Engineering Applications - Volume 2, Vasilios N. Katsikis, IntechOpen, DOI: 10.5772/46467. Available from: https://www.intechopen.com/chapters/39337
35. Wireless Sensor Network Simulation Frameworks: A Tutorial Review ... https://www.researchgate.net/publication/301273587_Wireless_Sensor_Network_Simulation_Frameworks_A_Tutorial_Review_MATLABSimulink_bests_the_rest.
36. “Throughput.” Wikipedia, Wikimedia Foundation, 22 Mar. 2022, https://en.wikipedia.org/wiki/Throughput.
37. “Download.” Wireshark · Go Deep., https://www.wireshark.org/.
38. “Ng: Nullnet.” Contiki, https://contiki-ng.readthedocs.io/en/master/_api/group__nullnet.html#details.
39. “Hardware Address.” Hardware Address - an Overview | ScienceDirect Topics, https://www.sciencedirect.com/topics/computer-science/hardware-address.
40. “Mac Address.” Wikipedia, Wikimedia Foundation, 3 May 2022, https://en.wikipedia.org/wiki/MAC_address.
41. “Main Page.” Zolertia RSS, http://wiki.zolertia.com/wiki/index.php/Main_Page.
42. A Secure Network Level Bridge for Wireless Sensor Networks. https://dias.library.tuc.gr/view/68659.
43. Zolertia Z1 Datasheet - Wiki.zolertia.com. http://wiki.zolertia.com/wiki/images/e/e8/Z1_RevC_Datasheet.pdf.
44. [Contiki-Developers] Emulated Motes CPU Clock Rate. https://contiki-developers.narkive.com/FCsZZvsv/emulated-motes-cpu-clock-rate.
45. Running and Testing Applications for Contiki OS Using Cooja Simulator. https://core.ac.uk/download/pdf/80817534.pdf.

46. Using Cooja for WSN Simulations: Some New Uses and Limits - Ewsn.org. https://www.ewsn.org/file-repository/ewsn2016/319_324_roussel.pdf.
47. IOT Emulation with Cooja - Wireless. http://wireless.ictp.it/school_2015/presentations/firstweek/ICTP-Cooja-Presentation-version0.pdf.
48. Medium Access Control Sublayer (Mac Sublayer). https://www.tutorialspoint.com/medium-access-control-sublayer-mac-sublayer.
49. Santos, Aldri L., et al. “Clustering and Reliability-Driven Mitigation of Routing Attacks in Massive IOT Systems - Journal of Internet Services and Applications.” SpringerOpen, Springer London, 6 Sept. 2019, https://jisajournal.springeropen.com/articles/10.1186/s13174-019-0117-8.
50. Terrainlos: An Outdoor Propagation Model for ... - Inrg.soe.ucsc.edu. https://inrg.soe.ucsc.edu/wp-content/uploads/2016/08/mascots_2016_terrainLOS.pdf.
51. “Doc: RFC 6550: RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks.” Hjp, https://www.hjp.at/doc/rfc/rfc6550.html.
52. “Examples/RPL-UDP · 4803E132CC922532DDC50652F071DBD67D32BEA4 · E3DA-Public / Contiki-Ng.” GitLab, https://gitlab.fbk.eu/e3da-public/contiki-ng/-/tree/4803e132cc922532ddc50652f071dbd67d32bea4/examples/rpl-udp.
53. Burke, John. “What Is Throughput? - Definition from Whatis.com.” SearchNetworking, TechTarget, 5 May 2015, https://www.techtarget.com/searchnetworking/definition/throughput.
54. “Performance Evaluation of the IEEE 802.15.4 Mac for Low-Rate Low-Power Wireless Networks.” IEEE Xplore, https://ieeexplore.ieee.org/document/1395158.
55. “MAC Protocols in Contikios.” MAC Protocols in ContikiOS - Contiki, https://anrg.usc.edu/contiki/index.php/MAC_protocols_in_ContikiOS.
56. Contiki-Os. “Contiki/Netstack.h at Master · Contiki-Os/Contiki.” GitHub, https://github.com/contiki-os/contiki/blob/master/core/net/netstack.h.
57. “Ng: Os/Net/Nullnet/Nullnet.h Source File.” Contiki, https://contiki-ng.readthedocs.io/en/master/_api/nullnet_8h_source.html.
58. Michel, Mathieu, and Bruno Quoitin. “Technical Report : Contikimac VS X-MAC Performance Analysis.” ArXiv.org, 6 Apr. 2016, https://arxiv.org/abs/1404.3589.
59. C. Pinola, "Evaluating the performance of synchronous and asynchronous media access control protocols in the contiki operating system," WORCESTER POLYTECHNIC INSTITUTE, 2012.
60. “NG: Os/Net/ipv6/Uip-nd6.c File Reference.” Contiki, https://contiki-ng.readthedocs.io/en/master/_api/uip-nd6_8c.html.
61. “Ng: UIP: The IPv6 Stack.” Contiki, https://contiki-ng.readthedocs.io/en/master/_api/group__uip.html.
62. “RFC 4861 - Neighbor Discovery for IP Version 6 (ipv6).” Document Search and Retrieval Page, https://datatracker.ietf.org/doc/html/rfc4861.
63. “Ng: Os/Net/Routing/Rpl-Classic/Rpl-Nbr-Policy.c Source File.” Contiki, https://contiki-ng.readthedocs.io/en/master/_api/rpl-classic_2rpl-nbr-policy_8c_source.html.
64. “Tutorial: IPv6 Ping - Contiki-Ng/Contiki-Ng Wiki.” About GitHub Wiki SEE - 420,000+ Wikis, Now Indexable., https://github-wiki-see.page/m/contiki-ng/contiki-ng/wiki/Tutorial%3A-IPv6-ping.
65. “Ng: Os/Net/ipv6/Uip-nd6.h Source File.” Contiki, https://contiki-ng.readthedocs.io/en/master/_api/uip-nd6_8h_source.html.
66. “Neighbor Discovery Protocol.” Wikipedia, Wikimedia Foundation, 4 Mar. 2022, https://en.wikipedia.org/wiki/Neighbor_Discovery_Protocol.

67. IPv6 Neighbor Discovery Protocol - System Administration Guide: IP Services. 1 Aug. 2011, https://docs.oracle.com/cd/E18752_01/html/816-4554/ipv6-ref-34.html.
68. “IPv6 Neighbor Discovery.” IPv6 Neighbor Discovery | Junos OS | Juniper Networks, https://www.juniper.net/documentation/us/en/software/junos/neighbor-discovery/topics/topic-map/ipv6-neighbor-discovery.html.
69. Neighbor Solicitation and Advertisement Messages. http://docs.ruckuswireless.com/fastiron/08.0.60/fastiron-08060-l3guide/GUID-4B7AE145-E81A-4826-8A28-9545490C374B.html.
70. Stehl´ık, Martin. “Comparison of Simulators for Wireless Sensor Networks.” Masters Thesis, https://is.muni.cz/.