Wmediumd for 802.15.4: Bridging the Gap Between Simulation and Physical Testbeds

1st Ramon dos Reis Fontes
Metropole Digital Institute
Federal University of Rio Grande do Norte
Natal, Brazil
ramon.fontes@ufrn.br

2rd Christian Esteve Rothenberg
Faculty of Electrical and Computer Engineering
University of Campinas
Campinas, Brazil
chesteve@unicamp.br

3nd Eduardo Cerqueira
Institute of Technology
Federal University of Pará
Pará, Brazil
cerqueira@ufpa.br

***Abstract*—Realistic emulation of wireless communication is essential for the development and validation of protocols in low-power networks. While tools like Wmediumd provide accurate medium emulation for IEEE 802.11, no equivalent exists for IEEE 802.15.4, an important standard underlying protocols such as ZigBee, Thread, and 6LoWPAN. This paper presents an extension of Wmediumd to support IEEE 802.15.4, enabling probabilistic modeling of wireless effects such as interference, attenuation, and packet loss. By integrating with the existing mac802154_hwsim kernel module, our approach allows for accurate, reproducible experimentation in virtual environments. We demonstrate the emulator's effectiveness through scenarios involving range-based communication and unstable links, highlighting its potential for advancing research and testing in IoT and low-power wireless networks.**

***Index Terms*—IEEE 802.15.4, wmediumd, realistic emulation**

I. Introduction

Wireless networking technologies are fundamental to modern digital infrastructure, particularly in IoT deployments that rely on low-power and resource-constrained devices. While protocols such as IEEE 802.11 (Wi-Fi) are well-supported by mature simulation and emulation tools, low-power standards like IEEE 802.15.4, which serve as the foundation for ZigBee, Thread, and 6LoWPAN, and others, often lack realistic, flexible test environments for system-level experimentation.

One of the core challenges in IEEE 802.15.4 networks is the need to operate under strict energy and processing constraints [1]. Devices typically rely on batteries and have limited computational capabilities, which makes it essential to optimize communication strategies. In this context, network emulation becomes critical, as it allows testing under realistic conditions, enabling evaluation of channel access mechanisms, transmission behavior, and power usage.

Another important factor is interference, as IEEE 802.15.4 operates in the crowded 2.4 GHz band, where it coexists with technologies such as Wi-Fi and household devices like microwave ovens [2]. Emulating interference and signal degradation in a controlled environment helps identify vulnerabilities and test mitigation strategies to improve reliability.

To address these challenges, this paper introduces wmediumd_802154, an extension of the wmediumd [3] wireless medium emulator originally designed for Wi-Fi, now adapted to support IEEE 802.15.4. Integrated with the mac802154_hwsim Linux driver, wmediumd_802154 enables real-time emulation of IEEE 802.15.4 networks with configurable interference modeling, loss patterns, and topology control. To evaluate our proposal, we present three experimental scenarios: (i) per-link loss probability model; (ii) per-link signal-to-noise ratio; and (iii) and an interference model that supports the definition of node positions.

The main contributions of this work are as follows: we (i) extend the mac802154_hwsim kernel module to support user-space communication through the Netlink interface, enabling integration with wmediumd_802154; (ii) implement interference modeling, signal degradation, and probabilistic packet delivery based on propagation parameters; (iii) design a modular and extensible framework for IEEE 802.15.4 medium emulation that mirrors the successful approach used in Wmediumd for IEEE 802.11; and (iv) validate the proposed solution through emulation experiments scenarios described above.

The remainder of this paper is structured as follows. Section 2 provides the background and foundational concepts; Section 3 reviews related work in the field; Section 4 presents the proposed system architecture, while Section 5 evaluates wmediumd_802154 in three diferente scenarios; Finally, Section 6 concludes the paper and outlines future directions.

II. Background

The development and validation of protocols for low-power wireless networks depend heavily on reliable testing environments that can emulate realistic conditions without the need for physical hardware. In this context, mac802154_hwsim plays a pivotal role as it provides a kernel-level simulation of IEEE 802.15.4 radio interfaces. By creating virtual transceivers that integrate seamlessly with the Linux mac802154 stack, this module enables researchers and developers to prototype, debug, and evaluate protocol behavior in a fully controlled and reproducible setting.

The architecture and operation of mac802154_hwsim revolve around its role as a kernel module that provides virtual IEEE 802.15.4 radios. When the module is loaded, it initializes a specified number of virtual radio interfaces, each simulating the behavior of a physical IEEE 802.15.4 transceiver. These interfaces are fully integrated with the Linux mac802154 stack, meaning they can be configured and managed using

978-3-903176-75-1 ©2025 IFIP

standard tools and layered with upper-layer protocols such as 6LoWPAN or RPL.

Internally, each virtual radio is treated by the kernel in the same way as a physical device. The radios can be configured independently with parameters like PAN ID, channel, and device addresses. When a frame is transmitted by one of these virtual radios, the module checks for other radios that are configured on the same channel. If such radios exist, the frame is directly forwarded to them, creating a basic point-to-multipoint communication model. However, this environment is idealized: there is no signal attenuation, interference, or packet loss modeled in this default setup. All transmissions are assumed to be successful if the radios share the same channel and PAN configuration.

To address this limitation and introduce realistic wireless behavior towards network emulation, mac802154_hwsim can be integrated with user-space applications. In this extended setup, when a radio transmits a frame, the module hands it off to an emulator via Netlink sockets. The emulator then applies probabilistic rules based on a model of the wireless medium, taking into account factors like signal strength, distance, and interference, before deciding whether the frame should be delivered, dropped, or corrupted.

Through this architecture, mac802154_hwsim enables the development and testing of low-power wireless network protocols entirely within a virtualized environment, with the ability to incorporate complex and dynamic conditions that resemble real-world deployments.

A. From Real Radios to Virtual Testing

The WLAN driver architecture in the Linux kernel follows a design pattern similar to other network device (netdev) drivers. Typically, wireless hardware connects to the system via interfaces such as PCI or USB, and the corresponding driver implementation depends on the type of interconnect used. From the perspective of higher network layers, a wireless netdev behaves similarly to a traditional Ethernet interface, but with added functionalities specific to wireless communication such as support for ad hoc modes.

Originally, network interface cards implemented most lower-layer protocols, including Medium Access Control (MAC), directly in hardware or embedded firmware. This approach limited flexibility for developers, as the proprietary nature of firmware restricted access and customization. To address these limitations, newer generations of wireless devices transitioned toward a software-based MAC architecture. In this model, known as SoftMAC, MAC functionalities are implemented in software and loaded as kernel modules, enabling greater flexibility and control.

However, accurately replicating real-world wireless conditions, such as interference, signal fading, and mobility-induced packet loss remains a major challenge. These phenomena typically require elaborate and costly physical testbeds, limiting the accessibility and scalability of experimental validation. To address this, the company Cozybit developed wmediumd[1], a userspace tool that emulates the wireless medium by introducing probabilistic transmission errors in virtualized environments (e.g., with Mininet-WiFi [4]).

B. Gaps in Emulation for Low-Power Wireless Protocols

While IEEE 802.11 dominates WLAN use cases, IEEE 802.15.4 is the foundation for many low-power wireless networks and IoT protocols, including ZigBee, Thread, and 6LoWPAN/RPL. This standard is designed for low energy consumption, low data rates (sufficient for sensors), highly dynamic mesh topologies, and others.

Just like 802.11, 802.15.4 networks are affected by interference, signal loss over distance, multipath effects, cross-technology interference, and mobility (e.g., in mobile sensor networks or drone-based monitoring). Therefore, realistic simulation/emulation tools are essential for developing and validating, routing protocols (e.g., RPL), energy-saving strategies, reliable communication for critical applications (e.g., disaster monitoring), and others.

Despite these needs, there is currently no widely adopted open-source tool that offers 802.15.4 emulation capabilities comparable to what Wmediumd provides for 802.11 networks. This lack of a realistic, configurable, and reproducible test environment for 802.15.4 creates a major gap in the research and development process.

Without such tools, researchers are often forced to rely on either simplified simulation environments, which may not reflect real-world dynamics, or costly and inflexible physical testbeds. As a result, evaluating protocol behavior under realistic wireless conditions remains a significant barrier, particularly for resource-constrained IoT devices where behavior is highly sensitive to environmental factors.

The extension of Wmediumd to support IEEE 802.15.4, as proposed in this work, addresses this critical gap by enabling software-defined control of wireless medium characteristics, including interference, signal strength, and packet loss, tailored for the unique constraints of low-power wireless networks.

III. Related Work

When comparing our implementation of wmediumd_802154 with other emulation platforms, few tools have contributed significantly to network experimentation. Notable examples include ns-3 [5], OMNeT++ [6], and Cooja, a simulator integrated with the Contiki OS [7]. Each of these solutions offers valuable features but also presents limitations when compared to the flexibility and realism provided by wmediumd_802154 combined with mac802154_hwsim.

ns-3 is a powerful and widely used network simulator, supporting detailed modeling of IEEE 802.15.4 MAC and PHY layers. It enables comprehensive analysis of low-power networks through customizable simulation scenarios. However, as a discrete-event simulator, it lacks real-time interaction capabilities and does not natively support full system-level execution. Additionally, it relies on custom APIs rather than

[1]https://github.com/cozybit/wmediumd

standard system calls, making it less suitable for integration with real-world applications, cryptographic libraries, or SDN experimentation.

OMNeT++ is another modular and extensible simulator, well-suited for simulating large-scale and heterogeneous IoT networks. It supports a wide range of communication protocols, including IEEE 802.15.4, and offers visual tools for debugging and analysis. Nonetheless, it shares ns-3's limitations regarding real-time execution and the absence of native SDN integration. Its simulation kernel is not designed for live interaction with Linux-based applications or user-space protocols.

Cooja, although historically important, has become increasingly outdated. Developed as part of the Contiki OS ecosystem, it is tailored for simulating sensor network firmware and supports protocols such as RPL and 6LoWPAN. While Cooja is effective for firmware-level validation and visualizing node behavior, reliance on emulated hardware platforms (e.g., MSP430), lack of support for standard Linux networking tools, and incompatibility with modern security stacks. Moreover, it does not support SDN integration or realistic experimentation with modern cryptographic techniques, making it less suitable for current IoT research involving system-level protocols or post-quantum security.

In contrast, mac802154_hwsim, integrated with our enhanced wmediumd_802154, offers a kernel-level emulation framework that enables real-time communication between virtual IEEE 802.15.4 interfaces. It supports actual Linux networking stacks, allowing experimentation with real applications, tools such as tcpdump and Wireshark, and libraries including OpenSSL or post-quantum cryptography frameworks. Furthermore, it can be easily extended and integrated with SDN tools (e.g., Open vSwitch, network namespaces), enabling programmable control over low-power networks, an aspect lacking in traditional emulators.

IV. WMEDIUMD 802.15.4

Figure 1 illustrates the architecture of Wmediumd 802.15.4, which allows realistic emulation of IEEE 802.15.4 wireless communication within the Linux operating system. At the core of this setup is the wmediumd 802.15.4 component, a user-space daemon responsible for simulating the behavior of the wireless medium, such as signal degradation, interference, and probabilistic frame loss. It interacts with the Linux kernel through the netlink interface, allowing it to control and monitor virtual radio interfaces.

The radios themselves are simulated by the mac802154_hwsim kernel module, which creates multiple virtual IEEE 802.15.4 radios (Radio #1 to Radio #n). These radios are grouped by communication channels (e.g., channel x), meaning that only radios on the same channel can potentially communicate with each other. Frames transmitted by one radio are intercepted by wmediumd_802154, which determines based on its internal model whether the frame should be delivered, dropped, or corrupted before forwarding it to the destination radio.

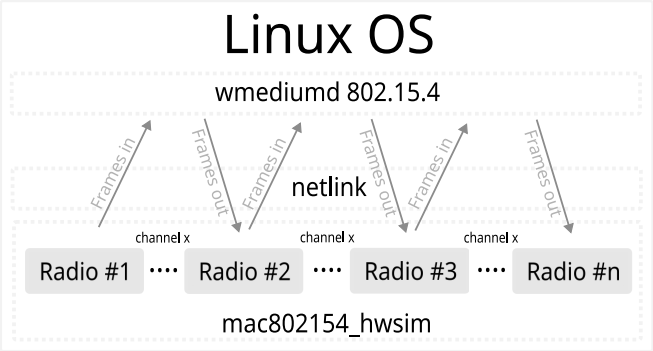

Fig. 1. Wmediumd 802.15.4 Architecture.

This architecture allows developers and researchers to emulate complex and dynamic wireless environments for IEEE 802.15.4 without the need for physical hardware, significantly improving the efficiency of testing and development for IoT and low-power wireless network protocols. Figure 2 illustrates the internal processing of frame transmission using mac802154_hwsim, integrating wmediumd_802154 logic for medium emulation. The dotted lines in the figure represent components or steps implemented or modified by us.

- **Frames In:** The process begins when a frame is received by the virtual radio interface implemented by mac802154_hwsim;
- **Transmission Handler:** The system invokes hwsim_hw_xmit() to handle the outgoing frame. At this point, the function checks whether the netlink communication (with Wmediumd) is active;
- If netlink is not running, the frame is passed directly to hwsim_hw_receive(), bypassing any emulation logic. Otherwise, the frame enters the Wmediumd emulation pipeline for further processing;
- **Frame Transmission Request:** The frame is then passed to mac802154_hwsim_tx_frame_nl(), which prepares it to be processed by Wmediumd;
- **Frame Processing:** Wmediumd receives the frame via netlink and calls process_recvd_data() to simulate realistic medium behavior;
- **Frame Delivery Decision** based on its internal medium model (signal strength, interference, distance, etc.), Wmediumd applies a probabilistic rule to determine whether the frame should be delivered via ap-

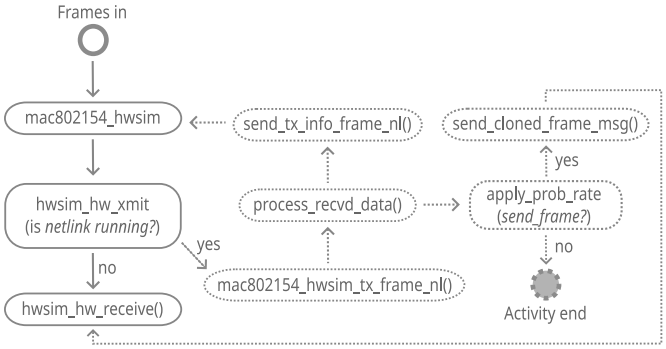

Fig. 2. Frames Processing Pipeline.

ply_prob_rate();
- If the frame is not to be delivered, the activity ends here. Otherwise, the process proceeds to send_cloned_frame_msg();
- **Transmission Acknowledgment:** report the transmission result back to the sender, completing the emulated delivery process with send_tx_info_frame_nl();
- **Frame Cloning and Delivery:** Finally, the frame is cloned and passed back into the virtual interface of the receiving radio via send_cloned_frame_msg().

A. Limitations

While wmediumd_802154 provides a practical and reproducible platform for emulating IEEE 802.15.4 networks, some limitations remain. First, although the emulator supports interference modeling and per-link SNR configuration, the current implementation does not yet incorporate time-varying or mobility-driven dynamics. The placement of nodes can be defined, but mobility itself (i.e., dynamic position updates and fast-changing channel conditions) is still an open direction.

Second, since emulation adds an additional software layer between the protocol stack and the hardware, there is a potential penalty in terms of computational overhead and resource usage. However, this impact is not critical for small to medium-sized network scenarios ($\sim$500 nodes), which can be executed even on modest computing resources - typically a modern multi-core CPU (e.g., 8 cores) and 16GB of RAM.

Despite these limitations, the results presented here already demonstrate that the emulator offers realistic and configurable communication costs, which support its main contribution. Mobility support is already being addressed through ongoing kernel contributions, and scalability experiments are planned as an extension of this work, leveraging the enhanced mobility features. In addition, future work will include hybrid experimentation (emulated + real radios) and energy-performance analysis to further strengthen the evaluation.

V. Evaluation

To validate the feasibility and effectiveness of wmediumd_802154, we conducted a series of emulation experiments using virtual IEEE 802.15.4 radios instantiated via our modified mac802154_hwsim module. This version incorporates support for Signal-to-Noise Ratio (SNR), packet loss, and an interference-aware propagation model. The goal of these experiments was to assess the impact of link-level configurations on packet delivery and to verify the correct operation of the medium emulation logic under realistic wireless conditions.

As illustrated in Figure 3 and for sake of simplicity, the evaluation environment comprises three virtual sensor nodes (`sensors` 0, 1, and 2) arranged in a tree network topology, with `Sensor0` serving as the coordinator node. Pairwise communication edges were manually configured between radios using the well known by the Linux kernel community wpan-tools[2], representing the connectivity matrix of the wireless medium.

[2]https://github.com/linux-wpan/wpan-tools/

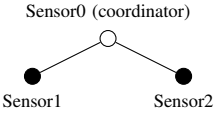

Fig. 3. Network Topology.

To facilitate reproducibility and enable further experimentation, all configuration files and scripts used in the evaluation are publicly available at https://github.com/ramonfontes/wmediumd_802154/.

A. Per-link loss probability model

Wmediumd_802154 provides a straightforward mechanism for emulating imperfect wireless communication by assigning fixed packet loss probabilities to individual links. This enables emulation of the wireless medium with deterministic loss behavior, offering a more realistic alternative to an idealized, error-free environment.

In this model, the user specifies a probability value for each link, representing the likelihood of packet loss, where values range from 0.0 (no loss) to 1.0 (100% loss). This configuration is particularly useful for creating reproducible and deterministic test scenarios involving lossy links or intentional connectivity gaps.

Each loss probability is defined in a link configuration block as a tuple in the format (*source, destination, error_prob*), where *error_prob* denotes the fixed packet drop rate for that specific link. Although this error model is static and does not account for dynamic factors such as signal degradation over distance, mobility, or varying data rates, it serves as a valuable tool for baseline testing, topology enforcement, and controlled scenario reproduction.

In our experiments, we successfully applied this configuration to emulate varying link qualities, demonstrating that wmediumd_802154 accurately enforces the specified packet delivery probabilities. For instance, when transmitting 10 ICMP packets, setting an *error_prob* of 0.2 (i.e., 20%) and 1.0 (i.e., 100%) resulted in approximately 20% and 100% packet loss, respectively.

B. Per-link signal-to-noise ratio (SNR) model

The per-link signal-to-noise ratio (SNR) model provides fine-grained control over the quality of wireless connections between virtual radios, allowing the emulation of asymmetric and heterogeneous network topologies. Low SNR values can significantly degrade or even block communication between nodes, accurately reflecting real-world scenarios where poor signal quality limits connectivity.

SNR values are also specified within a link configuration block. Each link is also defined as a tuple in the format (*source, destination, snr*), where *source* and *destination* correspond to the indices of virtual radios, and *snr* is a numerical value representing the signal-to-noise ratio.

In our experiments, we successfully applied this model to create controlled scenarios with both stable and unstable links. For instance, we configured the SNR value between `Sensor0` and `Sensor1` to 35 dB and between `Sensor1`

and `Sensor2` to 8 dB. As expected, communication between `Sensor0` and `Sensor1` remained stable, with near-zero packet loss, while communication with `Sensor2` exhibited frequent delivery failures and significant RTT variability due to the lower SNR.

Additionally, when the SNR was further reduced below 5 dB, packet delivery between `Sensor1` and `Sensor2` became virtually impossible, confirming that wmediumd_802154 reliably blocks transmissions in low-SNR environments. These results demonstrate the emulator's ability to reflect the asymmetric and lossy nature of real-world wireless channels, making it a valuable tool for testing the robustness of routing protocols and adaptive communication strategies in low-power IoT networks.

C. Interference

We now evaluate how the wmediumd_802154 handles interference and connectivity constraints based on link configuration, signal range, inter-node distance. Although wmediumd_802154 supports a variety of propagation models widely recognized in the literature, such as Friis, Log-Distance, ITU, Log-Normal Shadowing, among others, we chose to adopt the Log-Distance propagation model for our evaluation. By simulating realistic wireless conditions, the emulator enables validation of packet delivery behavior in scenarios involving communication loss, unstable links, and distance-induced degradation.

1) Out-of-Range Communication - No Link Established: This experiment evaluates a connectivity scenario involving spatial separation between sensor nodes. In this setup, `Sensor0` and `Sensor2` are intentionally positioned beyond each other's communication range, resulting in the absence of a direct link. Consequently, packet transmission attempts between these nodes result in loss, as the lack of overlap in their communication ranges prevents any successful delivery.

This experiment was conducted successfully and closely mirror those using the per-link SNR model. However, unlike the manually defined SNR values in the previous scenario, here the signal-to-noise ratio is automatically computed based on the relative position and distance between nodes, enabling dynamic and realistic calculation of signal attenuation.

This approach allows the emulator to determine link feasibility without requiring manual SNR input, making it particularly useful for scenarios involving node mobility or realistic spatial layouts. The observed 100% packet loss between `Sensor0` and `Sensor2` confirms that wmediumd_802154 accurately enforces communication constraints based on propagation-based SNR estimation.

2) Unstable Connectivity: This experiment investigates the behavior of the emulator under conditions in which nodes are technically within communication range, but link quality is impaired due to distance-induced attenuation. `Sensor0` maintains a stable connection with `Sensor1`, indicating adequate signal strength for consistent communication. In contrast, `Sensor2` is deliberately placed near the edge of the radio's effective range, resulting in an unreliable and unstable connection.

While occasional packet reception may occur due to marginal signal presence or leakage, the increased distance and reduced SNR lead to intermittent or failed transmissions. This behavior underscores the emulator's ability to capture and model the subtleties of real-world wireless degradation. The scenario is conceptually similar to the fixed probability loss model presented earlier; however, instead of defining delivery probability manually, here the signal-to-noise ratio (SNR) is calculated automatically based on the physical distance between nodes.

This outcome highlights how wmediumd_802154 dynamically adjusts delivery probability based on environmental factors, providing a realistic emulation of wireless behavior. The observed packet loss and RTT variability further reinforce the importance of modeling distance, interference, and signal degradation in the evaluation of IEEE 802.15.4-based low-power networks.

VI. Conclusion

This work presents wmediumd_802154, an extension of the Wmediumd framework integrating the mac802154_hwsim kernel module with probabilistic wireless medium models, enabling accurate emulation of IEEE 802.15.4 networks, including packet loss, interference, and signal degradation in both stable and degraded connectivity scenarios.

Future work will aim to enhance the interference models further by incorporating mobility, time-varying conditions, and environmental dynamics, increasing the realism and applicability of the emulator. Additionally, we envision extending the framework to support hybrid experimentation setups, allowing integration with physical IEEE 802.15.4 devices. This would enable mixed virtual/real testbeds, improving the fidelity of experiments while maintaining control and scalability.

Acknowledgment

This research was partially sponsored by CAPES grant Process #88887.005666/2024-00 and FAPESP #21/00199-8.

References

[1] A. Adeel *et al.*, "A survey on the role of wireless sensor networks and iot in disaster management," *Geological disaster monitoring based on sensor networks*, pp. 57–66, 2019.

[2] R. Hassan, M. Rahman, and M. Shakir, "Performance analysis of ieee 802.15.4 for wireless sensor networks," *International Journal of Computer Applications*, vol. 175, no. 6, pp. 30–35, 2017.

[3] A. Martínez Illán, "Medium and mobility behaviour insertion for 802.11 emulated networks," Master's thesis, Universitat Politècnica de Catalunya, 2013.

[4] R. R. Fontes *et al.*, "Mininet-wifi: Emulating software-defined wireless networks," in *2015 11th International conference on network and service management (CNSM)*, pp. 384–389, IEEE, 2015.

[5] ns 3, "ns-3: Discrete-event network simulator," 2024. Accessed: 2025-06-05.

[6] OMNeT++, "Omnet++: Discrete event simulator," 2024. Accessed: 2025-06-05.

[7] C. OS, "Contiki: The open source operating system for the internet of things," 2024. Accessed: 2025-06-05.