SEmulate: Seamless Network Protocol Simulation and Radio Channel Emulation for Wireless Sensor Networks

Sebastian Boehm and Hartmut Koenig
Brandenburg University of Technology Cottbus–Senftenberg
Cottbus, Germany
eMail: {sebastian.boehm, hartmut.koenig}@b-tu.de

***Abstract*—Low-rate and low-power wireless communications are still the main drivers for innovative industrial automation and the Internet of Things (IoT). Physical mobility is one of the most important challenges for them. Common wireless technologies and protocols, e.g., WirelessHART and ISA100.11a Wireless for industrial process plants, ZigBee for building automation, or 6LoWPAN and 6TiSCH in context of the IoT, are based on the IEEE 802.15.4 standard. Event-based simulation is the method of choice for analyzing network protocols and algorithmic applications of such distributed sensor applications. Performance measurements and holistic evaluations, however, are greatly influenced by the underlying hardware resources, physical layer protocols, and radio channel conditions, which are usually not considered or highly abstracted in network simulations. In this paper we present SEmulate, a hybrid system for seamless (network) simulation and hardware-based emulation for wireless sensor networks based on the IEEE 802.15.4 protocol standard, which takes the hardware aspects into account by applying an Hardware-in-the-Loop (HIL) approach.**

***Index Terms*—Hardware-in-the-loop, Radio-in-the-Loop, simulation, emulation, wireless sensor networks, cross-layering, co-simulation, OMNeT++**

I. Introduction

The Internet of Things (IoT) and embedded wireless devices are very specific in terms of hardware, radio technologies, and network protocols. From the perspective of (wireless) communication protocol architectures (and apart from the proprietary hardware software solutions that still exist), the 802.15.4 specification of the Institute of Electrical and Electronics Engineers (IEEE) is the predominant standard. It constitutes the basis for wireless network access in industrial process plants, wireless building automation, and consumer electronics. Precisely, common and widely used technologies and protocols like WirelessHART, ISA100.11a, ZigBee, 6LoWPAN, or 6TiSCH are based on this standard.

Traditional evaluation methods for wireless communication architectures and protocols are usually based on simulation techniques. The network simulation approach simplifies the construction of layered communication protocol architectures and application models for networked systems. A special problem represents the simulation of the lower layers, in particular the Physical Layer (PHY) because it is still difficult to model all hardware-related aspects and details of the specification. Models for the PHY and the physical radio channel conditions usually base on abstracted mathematical functions of varying complexity. In wireless systems the PHY is a very complex layer that can be divided into several domains (e.g., bits, symbols, samples, analog waveforms). These different domains are too complex to model in event-based simulation systems. The radio channel and communication side effects are typically simplified or completely omitted in such simulations.

Radio Channel Emulation is an approach that connects the physical radio antenna ports of the devices to an emulator testbed. The radio channel is then reproduced with the help of analogous Radio Frequency (RF) hardware or Digital Signal Processors (DSPs) in a laboratory setup in real-time. In this context, emulation allows us to manipulate and reproduce wireless channel conditions like signal fading, interferences, and noise. Normally, the emulated channel properties can be adjusted at run time by means of software. Since real devices and radio hardware are used to transmit and receive RF signals, the radio channel emulation provides a much more realistic picture than traditional network simulation can do. The drawback is that network protocols and applications are fixed in custom-developed device firmware which do not provide any degrees of freedom for experimenting with different protocols, communication architectures, or parameters.

In this paper we present SEmulate, a system for a seamless (network) simulation and hardware-based emulation of Wireless Sensor Networks (WSNs) based on the IEEE 802.15.4 standard. For this hybrid evaluation system, we apply a Hardware-in-the-Loop simulation setup, as it is widely deployed in automotive industry. In contrast to the latter, which mainly tests single devices, we connect several hardware devices in an emulated environment to support a common generic and high-level network protocol simulation.

Our hybrid approach leverages on the strengths and benefits of the individual approaches to enable precise and likewise flexible tests for WSNs. Thus, we are able to study the effects of varying radio channel conditions and network topologies on the whole network, as well as the protocol behavior of single wireless sensor nodes which are difficult to reproduce in real testbeds. Furthermore, we provide actual measured values of the PHY (e.g., link quality, peak channel energy) for further processing on higher protocol layers to simulate WSN cross-layer approaches.

ISBN 978-3-903176-13-3

The remainder of the paper is structured as follows. Section II gives a short overview about related work regarding WSN and IoT protocol simulation, radio channel emulation, and HIL simulation approaches. In Section III we introduce our simulation and emulation concept *SEmulate* as a solution for a precise evaluation of wireless systems by means of HIL. In the subsequent sections we detail the three main components of the SEmulate system architecture. In Section IV we describe the simulator implementation. Section V presents the stream forwarder. Next in Section VI we consider the emulation hardware. Finally in Section VII, we present some selected evaluations of parts of the system. We conclude the paper with some final remarks and give a short outlook on the next research steps.

II. Related Work

A. WSN and IoT Protocol Simulation

Event-based network protocol simulators are based on software building blocks that model the communication protocol stack. A popular open source Discrete Event Simulation (DES) simulator is OMNeT++, which is well-known in academia. OMNeT++ has a great and active community, as well as several add-on frameworks for protocol simulation like INET. Simulation models for common communication protocols of the network layer and especially the upper layers of the communication protocol stack are widely available [1]. Regarding the Data Link Layer (DLL) and in particular the Physical Layer, the details of available models for wireless transmission systems and their specific focus vary greatly. In addition, modern wireless protocol standards like IEEE 802.15.4 typically support a multitude of alternative PHYs for various application-specific use cases. Modeling all these PHYs with their complex internals and modulation schemes represents a huge challenge.

Looking at the physical process modeling domain, there are precise simulation modules and evaluations for IEEE 802.15.4 PHY [2] in Matlab[1] in which different modulation techniques (e.g., QPSK, BPSK, ASK, GFSK) and the resulting waveform generation are modeled. In general, only a few communication protocols are modeled in Matlab. As consequence, Matlab is not able to provide a holistic evaluation of WSNs and communication systems.

With the new IEEE 802.15.4 model [3] for OMNeT++ / INET, a model has been designed that is very close to the original standard in terms of the complex and versatile Medium Access Control (MAC) layer functionality and the general PHY requirements and services. We modeled the addressed two layers with all connection interfaces, service primitives, and all Protocol Data Unit (PDU) definitions according to the general modeling guidelines for IEEE 802.15.4 [4]. Some common modulation techniques though (e.g., QPSK) are not integrated neither in the 802.15.4 model nor in the OMNeT++ / INET's radio models, yet.

[1]End-to-End IEEE 802.15.4 PHY Simulation in Matlab: https://www.mathworks.com/help/supportpkg/zigbee/examples/end-to-end-ieee-802-15-4-phy-simulation.html

Another pretty accurate but functionally limited IEEE 802.15.4 simulation model is presented by Amin & Abdel-Hamid [5]. Its particular focus relies on the Guaranteed Time Slot (GTS) mechanism of the MAC layer. It has been built to demonstrate GTS attacks. The model does not implement all MAC-specific PDUs and MAC layer specifications (e.g., the CSMA-CA protocol), but only GTS-relevant features. Furthermore, the PHY is not implemented at all in this model because it is not needed for demonstrating GTS attacks.

Other IEEE 802.15.4 simulation models have already been referred to in [3] and [5], respectively. To sum up, there is no all-embracing IEEE 802.15.4 model for WSN simulation up to now.

B. Radio Channel Emulation

Experience with the evaluation of the radio channel for a small number of nodes is reported in [6]. The authors set up a two-hop relay network in which the actual sensor node hardware is packed into RF-shielded boxes and attached by means of coax cables and analogous RF hardware to an emulation environment with a two-channel wireless emulator. Thus, they can avoid direct transmissions from the source to the sink. In addition, they added a controllable RF environment for tests using different fading characteristics among the nodes. As a result, they could demonstrate, how such a testbed can obtain and repeat the performance setup for a particular channel environment. This evaluation method, however, is laboratory-based and very specific to the target application scenario.

Beshay et al. [7] demonstrate an Field Programmable Gate Array (FPGA)-based emulation system that allows more complex wireless experiments. This system emulates the signal propagation, fading, bidirectional links, and node mobility in custom topologies. With so-called multi-tap fading generators, they are able to use user-defined parameters for signal fading and time delay of multiple channels for repeating experiments under identical channel conditions. The tests were performed with two wireless nodes, but the authors proclaim a potential scalability of the system up to hundreds of nodes. Systems that emulate the propagation of radio signals based on a FPGA are usually called hardware channel simulators.

There are a few commercial solutions for channel emulation available[2][3]. However, these systems are very expensive, partially obscure, and limited in the number of networked nodes. Continuing more precisely, these systems are not suitable for scalable experiments with interactions (and interferences) among multiple nodes with a high degree of connectivity.

C. HIL & Co-Simulation

Hardware-in-the-Loop (HIL) simulation approaches are practically established in automotive or robotic application domains. In these areas HIL approaches are used to verify existing hardware module implementations. Normally, a

[2]RFnest™: https://www.i-a-i.com/product/rfnest/

[3]PROPSIM Channel Emulation Solutions: https://www.keysight.com/en/pc-2697334/propsim-channel-emulation-solutions

ISBN 978-3-903176-13-3

simulator provides a virtual environment and is generating input for the connected hardware Device Under Test (DUT). In the area of embedded wireless systems, in contrast, HIL simulation approaches and related methods are rarely applied (cp. [8]) and, if so, they often focus on a specific application-domain like cross-platform development [9] and automatic code generation [10].

In the area of vehicular networking, there is a hybrid system concept in which a Vehicular Ad Hoc Network (VANET) simulation is combined with real-time systems in HIL fashion [11]. In this very specialized hybrid evaluation system concept for large-scale VANET scenarios, discrete-event simulation and real-time hardware are integrated to run concurrently.

Staub et al. [12] propose an emulation framework for IEEE 802.11 ad-hoc networks in OMNeT++. With VirtualMesh, they integrate real nodes with virtual interfaces and fully virtualized nodes into a single simulated environment. The network stack of real and virtualized nodes is subdivided into two different domains. The real nodes correspond to real device hardware that is set up to generate real traffic on the application, transport, and Internet layer. The MAC layer and the physical medium of all nodes as well as the network topology including physical effects are entirely simulated.

The emulation system of Weingärtner et al. [13] follows a similar approach for IEEE 802.11. They integrate instances of custom operating system drivers for wireless devices to enable the simulator-integration of real-world network software. Traffic among simulated nodes and the device driver is exchanged with so-called gateway nodes at the MAC layer. Fully simulated nodes, in contrast, provide the entire protocol stack in the simulation. Furthermore, the simulation models the MAC and the PHY layers, as well as the wireless channel and virtual node mobility. They enable real-world wireless networking software investigation inside a simulation-controlled environment.

Wehner & Goehringer [14] present a HIL coupling concept that supports a real-time message exchange with external hardware interfaces. They provide a gateway functionality in OMNeT++ that basically enables a message forwarding between different real-life devices. The primary focus of their approach is to translate PDUs of different proprietary communication technologies, e.g., EnOcean and Z-Wave. They do not consider any simulation of higher protocol layers and no simulation or emulation of the radio channel.

We presented our first experiences of bridging network simulation and real device hardware to increase the evaluation accuracy of the lower layers of the protocol stack in [15]. After starting with a simple test setup with OMNeT++ / INET's *PCAPRecorder*, we continued developing a holistic IEEE 802.15.4 WSN evaluation approach. Thereafter in [16], we have introduced a basic methodology for improving simulation-driven wireless sensor network evaluations by means of HIL.

The following sections gives a detailed overview of the further development of our methodology with the SEmulate emulation system concept.

III. The SEmulate system

The basic methodology of the SEmulate system follows the layered protocol architecture in networked communication systems. It applies a multidisciplinary concept for coupling network protocol simulation with radio channel emulation using radio Hardware-in-the-Loop. Figure 1 shows the protocol stack and the interaction with the communication medium of an IEEE 802.15.4 based sensor node to which we refer below.

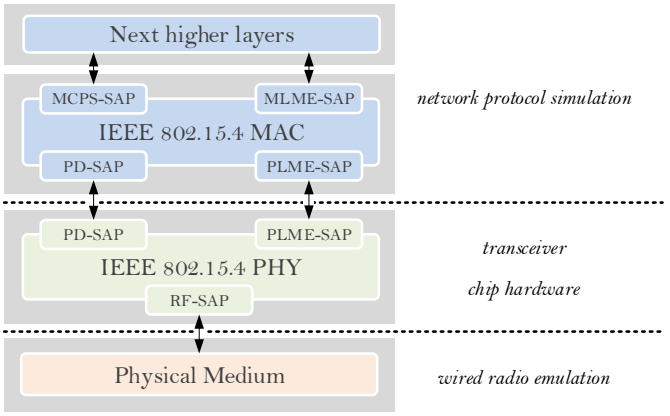

Fig. 1. Node Communication Model Architecture

A. Coupling Concept

The protocol stack splits in two parts. The respective protocol layers are assigned to different simulation and emulation evaluation domains. We have placed the hardware-independent layers in a pure event-based simulation domain because logical protocol sequences and algorithmic procedures can be readily modeled with defined system states and state changes over time. For the simulation of the higher layers of the protocol stack, a great variety of protocols and models is available in OMNeT++ / INET, which we do not explicitly introduce here. The IEEE 802.15.4 MAC layer is also placed in the simulation domain. It defines all mechanisms and features for segmentation and reassembly, queuing, and error detection of data frames. It also cooperates though with the PHY layer in the time domain for multiple access or radio resource management, which is often highly abstracted or completely absent in pure simulations.

Referring to Figure 1, we share all PHY-relevant accesses from the MAC layer for data and management service functionality with the execution on the real transceiver chip hardware. The radio chip on the target hardware can send/receive data frames on/from the radio channel and to transfer accurate measurements of the actual channel state to the simulated MAC. Using real hardware on the PHY layer mitigates the major drawbacks of abstracted physical measurements in typical WSN simulations.

We use radio channel emulation to overcome the disadvantages of hard-to-predict radio environments which are common in regular wireless testbeds. By means of the interactions with the PHY layer, the MAC, and the higher layers, we are able to

ISBN 978-3-903176-13-3

observe protocol behavior under different channel conditions. Furthermore, we lay the foundation for accurate cross-layer optimizations of protocols, functions, and applications that refer to the actual, fine-grained radio state in a repeatable and controllable physical environment.

The type of the system coupling of [11] introduces a comparable coupling concept, but is not applicable to our methodology, as it defines very concrete interfaces in a completely different context. The approaches of Staub et al. [12] and Weingärtner et al. [13] demonstrate how to generate traffic from the upper network layers of real devices for a MAC and PHY layer simulation. This is exactly in contrast to our methodology of combining the simulation of higher layer protocols with accurate PHY and radio emulation.

Our approach of combining simulation and radio channel emulation with a HIL concept for WSN testing is called Radio-in-the-Loop (RIL) in the following.

B. System Architecture

The SEmulate architecture is depicted in Figure 2. It consists of two subsystems - the simulation and the emulation domain - and an intermediate component - the stream forwarder. In the simulation domain we set up OMNeT++ with

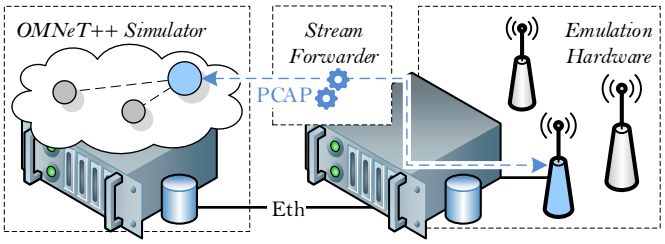

Fig. 2. SEmulate RIL System Architecture

customized models for IEEE 802.15.4 and interfaces to the emulation domain. We simulate the MAC layer and the higher layers of the protocol stack in OMNeT++ (see network protocol simulation in Figure 1). The emulation domain is provided by the SEmulate control and hardware subsystem. The PHY layer at hardware-level and the wired radio emulation of our methodology are realized in this domain. The modular stream forwarder forms the link between the two domains. Thus, we provide a generic stream processing system that is able to bridge between different and interchangeable subsystems, the simulator OMNeT++ and the SEmulate hardware system in our case.

In order to support the interoperability among the architecture's components there is a strong need for a consistent semantics and data interpretation in all involved subsystems. Within the computer network community, *PcapNG* (formerly: *PCAP*) is considered as a de facto standard for network packet readings from a Network Interface Controller (NIC). We have chosen it as control and data exchange protocol among all involved subsystems, not least because it supports the individual extensibility of frame formats with embedded optional fields and the capability of carrying data frames from multiple network interfaces within one single data stream. We break with the regular usage that PcapNG streams only carry DLL frames. For the internal message processing and handling, we have implemented a subset of PcapNG, the three basic block types that are relevant for our use case:

- Section Header Block (SHB) *(init PcapNG stream)*
- Interface Description Block (IDB) *(set device interface)*
- Enhanced Packet Block (EPB) *(process packet)*

IV. RIL Simulator Implementation

In order to make the simulated nodes capable to connect to real sensor nodes on the emulation hardware platform, we have added some emulation features and interface modules to the OMNeT++ IEEE 802.15.4 simulation model [3], as mentioned in subsection II-A. In the following we give a detailed overview of our simulation model architecture and its modules. Figure 3 depicts the emulation and interface modules as well as the event and information flows from a simulated node to the host system. It serves as basis for the further explanation of our RIL simulator implementation.

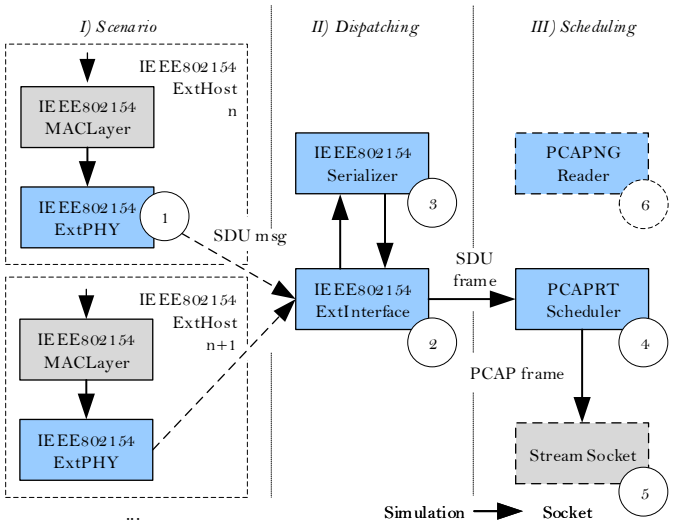

Fig. 3. IEEE 802.15.4 simulation modules in OMNeT++ / INET

A. IEEE 802.15.4 Simulation Modules

I) Scenario: For the emulation mode, we created the `IEEE802154ExtHost` which acts as representation of a sensor node module in the hardware system. This module is derived from the standard wireless host of our simulation model [3], as mentioned in Section II, but it has no radio interface and is not able to communicate with a virtual/simulated wireless channel. Instead of a PHY layer simulation module with gates to a radio interface, we apply an `IEEE802154ExtPHY` module (ref. ① in Figure 3) that models the handling and transfer of all management and data services standardized in IEEE 802.15.4 to the external PHY. The `IEEE802154ExtPHY` module can be considered as an abstract PHY with no IEEE 802.15.4-related functionality within the simulation.

II) Dispatching: The communication of the virtual nodes in the simulation scenario is transferred via the `IEEE802154ExtInterface` module (ref. ② in Figure 3). This dispatcher module adds an identification of the node and serializes the internal message representation of the simulation module to an ordinary byte format (ref. ③ in Figure 3) which is transferred for processing to the scheduler then. We have modeled all specific MAC PDUs and created respective serializer functions to convert these two message representations into each other. Since there is no specification in the IEEE standard of how PHY Service Data Units (SDUs) are represented internally, we have added a simple byte format (see Figure 4 for an example PHY SDU). In the case of

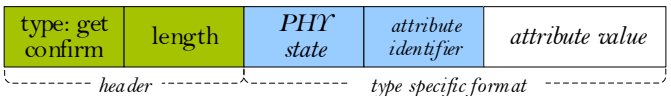

Fig. 4. Example IEEE 802.15.4 'PLME GetConfirm' frame format

the PHY data service, the encapsulated MAC frame is also serialized. Hence, the serializer module is responsible for converting between the two message representations of PHY SDUs and MAC PDUs. For the encapsulated higher layer protocols, we also need to call or add additional higher layer protocol serializers that should be provided by the higher layer simulation models.

B. Simulator Host Interface and Runtime Environment

The basic concept for connecting an OMNeT++ simulation with a host system was first published in [17]. We have adapted and extended this approach to enable the RIL-related node operations over the PcapNG frame format. Our implementation focuses on stream sockets (ref. ⑤ in Figure 3) for sending and receiving packets to and from the *outside* world.

III) Scheduling: The *PACP Real-Time Scheduler* (ref. ④ in Figure 3) is a custom stream socket scheduler that focuses on the transmission and reception of PcapNG data streams via a socket connection. Our first experiments are based on a Transmission Control Protocol (TCP) socket implementation. Since the simulator and the stream forwarder are running on the same Linux host system in our case, we also implemented and primarily use Unix Domain Sockets (UDSs). UDS communication is faster than TCP communication on the local host system due to the reduced protocol and network driver overhead.

The socket transmission of SDUs from the dispatcher simulation module is very straight-forward. We generate the specific PcapNG header fields and send out the resulting frame. For the return path of the messages, we have to identify the PcapNG frames from the socket byte stream in the receiving process. As PcapNG frames do not necessarily have to arrive completely with current bytes read, the scheduler incorporates in the `PCAPNGReader` (ref. ⑥ in Figure 3) which builds the resulting frames from stream segments.

V. Stream Forwarder

The *Stream Forwarder* is the intermediate system between the simulation and emulation domain. The multi-threaded application processes PcapNG frames to and from multiple inputs and outputs and enables the manipulation and display of the exchanged PcapNG frames.

This approach allows us to connect between the different evaluation systems. We use in-band signaling for the synchronization based on the PcapNG frame format. PcapNG can handle multiple interfaces (IDBs) with own link layer protocols and options. Hence, our application can aggregate or filter data from different end devices to compose a single data stream for further processing in the simulation system. The stream forwarder has a modular structure with the following connection interfaces on Linux systems:

- Transmission Control Protocol (TCP),
- Unix Domain Socket (UDS),
- Representational State Transfer (REST),
- Serial Port Terminal, and
- Pseudo Terminal *(for virtual device hardware)*.

A great advance of this solution is that the simulation system does not need any information about the emulation domain or implement possibly specific interfaces. Figure 5 depicts a possible setup as a *Terminal Forwarder* with one simulation interface and multiple connected hardware devices.

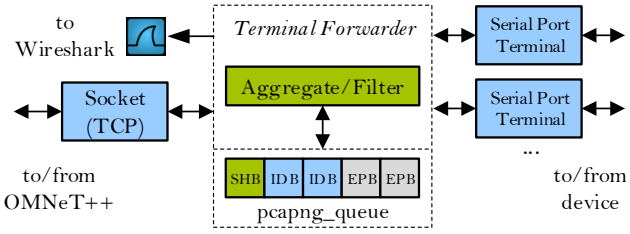

Fig. 5. Example of a Forwarder setup with multiple serial devices

The stream forwarder is not allowed to lose any packets that arrive at the various interfaces at the same time. Therefore, we have implemented every single interface connection and the central forwarder logic as separate threads. As a common data structure for processing the PcapNG data of different interfaces, we have implemented a thread-safe queue. For example, in order to consolidate packets from different hardware instances into a single output stream to the simulator, we manipulate each packet to ensure the correct interface assignment of each individual PcapNG frame of this hardware. Vice versa, we have to dispatch all packets from the simulation domain to the respective hardware devices based on the interface identifier.

Furthermore, we have implemented an interface to the de facto standard packet analyzer *Wireshark* (ref. Figure 5). Thus, we are able to send the PcapNG frames at run time to the standard input of the Wireshark application for monitoring and further processing of the transmitted streams. A custom packet dissector visualizes the IEEE 802.15.4 PHY SDUs for analyzing and debugging purposes (ref. Figure 6).

ISBN 978-3-903176-13-3

```
Frame 15: 4 bytes captured (32 bits) on interface 1
IEEE 802.15.4 PSDU
    Type: 0x0b (PLME_SET_REQUEST)
    Length: 4
  PLME_SET_REQUEST
      Attribute Identifier: 0x02 (phyTransmitPower)
      Value: 0x03 (Tolerance: +-1dB, Transmit Power: -16dBm)
```

Fig. 6. IEEE 802.15.4 SDU dissector in Wireshark

VI. Emulation Hardware

As the control and hardware subsystem is very complex and extensive, we only focus our explanations on details about the emulation architecture and its features, in particular the simulation/emulation firmware of the nodes and the hardware/emulation system architecture. We omit details about the control, processing, and management components here, as they are not relevant in the context of the emulation procedure.

A. Simulation/Emulation Node Firmware

To enable the inter-operation with real transceiver chip hardware including the transmission and receiving of real radio packets we have extended the WSN operating system Contiki [18]. We have implemented the transmission and receiving of PcapNG protocol frames via a serial interface to process the encapsulated PHY SDU messages in the chip hardware and to eventually access the physical radio channel on the emulated environment. The pseudo-code in Algorithm 1 illustrates the transceiver event processing, derived from our current Contiki-based implementation. The required *promiscuous mode* enables a network interface to pass all network traffic captured from the medium to the host system. Hence, we have introduced the event processing of PcapNG frames and IEEE 802.15.4 PHY SDUs for the operation with radio driver software in Contiki. The implementation is currently running on sensor nodes with ATmega128RFA1 radios[4], but the simulation/emulation firmware can also used for other hardware platforms provided in Contiki with only tiny adjustments regarding the serial interface configuration and promiscuous mode activation on the device.

Algorithm 1: Nodes Transceiver Process

```
Precondition : Init radio driver in promiscuous mode
Function receive()                    ▷ radio driver callback
    generate DATA_indication PHY SDU
    serialize and send indication via PcapNG
return
while (true) do
    wait until PcapNG serial interface event occur
    deserialize encapsulated request PHY SDU
    Function handle_Message(&msg)
        decode/exec cmd               ▷ e.g., driver operation
        generate confirmation PHY SDU
        serialize and send confirmation via PcapNG
    return
end
```

[4]ATmega128RFA1: https://www.microchip.com/ATmega128RFA1

B. Hardware System Architecture

Based on the architectural design, the hardware subsystem is able to emulate up to 1000 sensor nodes. The basic hardware component is an *Emulation Panel* that can be equipped with up to eight wireless sensor nodes on intended slots. The interconnection of panels and the configuration assembly of the nodes are variable, but a panel can also operate seperately. Multiple panels in the SEmulate emulation hardware system are (apart from the power supply) connected via a control and a RF network.

Control Network: The hardware control network is based on a switched Ethernet network which connects all emulation panels of the system. All commands, both the RF settings for the wired emulation as well as the panel, and node control are enabled via a controller on each panel. They are dispatched and processed at the central control server which communicates with each panel via a separate TCP connection. For example, the `SendTransparentData` command primitive is used to transmit the PHY SDUs to the nodes on the panels. In turn, the command `cmdSetAttenuation` enables the adjustment of the signal attenuation within the RF network which will be explained below.

RF Network: The RF network is based on the combination of high-frequency components and wireless sensor nodes on the panels via coaxial cables. Figure 7 represents one single emulation panel as undirected graph. Each additional panel connects to the output of the predecessor panel via its own input (see *P3* and *P0* in Figure 7). The weights of the edges represent the signal attenuation on each link. The RF network has both fixed (non-adjustable) and variable (adjustable) signal attenuations in the signal path of the coaxial environment. Hence, we can adjust the signal attenuation at each output path of a panel and thus between groups of nodes.

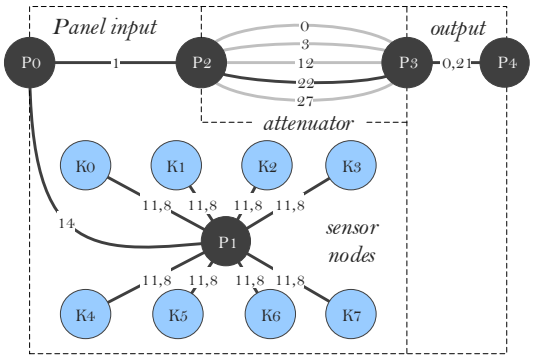

Fig. 7. RF network representation of one emulation panel *(P4 corresponds to the input from another panel)*. With the attenuator module, we can adjust the signal attenuation between adjacent panels.

In order to reproduce a (virtual) WSN with nodes at the coordinates in a concrete radio topology on the emulation hardware, we use existing models of signal propagation and

fading [19]. For the resulting signal attenuation values between pairs of nodes, the appropriate panels and sensor node slots on the emulation hardware as well as the values for the respective signal attenuators must be assigned. We already presented initial solutions and a basic algorithm for this problem by means of linear optimization in [16].

VII. Evaluation

In order to be used as an evaluation system the SEmulate system must meet the requirements of implementation accuracy. Furthermore, it has to ensure requirements for different kinds of system performance. As the implementation of SEmulate is still ongoing, we present first results of accuracy aspects and some evaluations of submodules in this chapter.

A. RIL Scenario

For a small-scale scenario, we performed tests covering the entire test track from simulation via the forwarder to (in this case virtual) device hardware. In our simulation we defined a small test setup of two IEEE 802.15.4 nodes competing to start a Wireless Personal Area Network (WPAN). The forwarder application was configured in a way so that it could connect to OMNeT++ via UDS and to the virtual device hardware via pseudo terminals. In this purely functional evaluation we were able to verify the correct protocol message sequences of the IEEE 802.15.4 protocol between the simulated MAC and emulated PHY layer and thus the accuracy of the procedure (see Figure 8 and Figure 84–Data transmission message sequence chart in [20]).

Event	Time	Src/Dest	Name
#20	0.526108	IEEE802154Nodes[1] --> IEEE802154Ext	PLME-SET-TRX-STATE.request
#22	0.526154	IEEE802154Ext --> IEEE802154Nodes[1]	PLME-SET-TRX-STATE.confirm
#25	0.526154	IEEE802154Nodes[1] --> IEEE802154Ext	PLME-CCA.request
#27	0.526216	IEEE802154Ext --> IEEE802154Nodes[1]	PLME-CCA.confirm
#30	0.526216	IEEE802154Nodes[1] --> IEEE802154Ext	PLME-SET-TRX-STATE.request
#32	0.526255	IEEE802154Ext --> IEEE802154Nodes[1]	PLME-SET-TRX-STATE.confirm
#35	0.526255	IEEE802154Nodes[1] --> IEEE802154Ext	PD-DATA
#37	0.526332	IEEE802154Ext --> IEEE802154Nodes[1]	PD-DATA.confirm
#40	0.526332	IEEE802154Nodes[1] --> IEEE802154Ext	PLME-SET-TRX-STATE.request
#42	0.526378	IEEE802154Ext --> IEEE802154Nodes[1]	PLME-SET-TRX-STATE.confirm

Fig. 8. IEEE 802.15.4 data transmission message sequence in OMNeT++

B. Stream Forwarder

In terms of throughput, the forwarder has to ensure a transparent reception, processing, and transmission of PcapNG frames. We carried out some performance measurements by a stress test of the forwarder application. For example, in one test setup, the forwarder connected to two test applications via a TCP interface each. For the transmission of maximum sized IEEE 802.15.4 data frames between the two test applications (10k data frames with 127 bytes each – overall data amount of 1.52 MB), we measured a TCP throughput of 91.5 Mbit/s from the sender test application to the forwarder and 91.4 Mbit/s from the sender to the receiver test application with 0% packet loss.[5] Since these measurements show a 0.11% TCP throughput decrease caused by the forwarder, we assume that our threaded interfaces and packet handling routines do not have a significant influence on the overall throughput.

[5]We performed our measurements on an Ubuntu Linux (64 bit) virtual machine with 7 vCPUs (Intel Xeon X3470 Quad Core @ 2.93 GHz).

C. Emulation Hardware

In order to assess the attenuation aspect of the emulation hardware we performed some Received Signal Strength Indication (RSSI) measurements to determine the actual signal attenuation values for the different step attenuator values, as depicted in Figure 7. In each case, the signal attenuation was

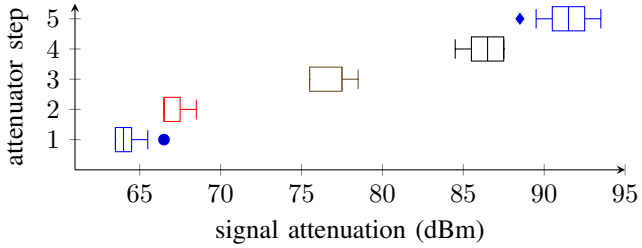

Fig. 9. Attenuation measurements

determined for pairs of nodes on adjacent panels. As we can see in Figure 9, the majority of the measured values are within a tolerance range of 2-3 dBm. Thus, we are able to emulate large-scale fading in WSNs.

VIII. Conclusions

With SEmulate, we have introduced a new tool chain for pure protocol simulation with an emulation concept for the PHY layer and the physical radio channel. Our approach allows the protocol simulation to execute all procedures and features defined in the IEEE 802.15.4 standard on real target hardware and provide valid RF measurements for the simulated MAC layer to improve higher layer protocol behavior and cross-layer studies under more realistic and reproducible radio conditions.

Nevertheless, SEmulate still requires further evaluations regarding performance aspects in large-scale scenarios and the feature set of the channel emulation domain. As next step, we will roll out large-scale emulation scenarios and plan to improve and extend the implementation of individual system components to support simulations of more complex WSN scenarios. After these extensive tests it will also be possible to provide more accurate details on the runtime and delays in the system. We will improve and conclude the automated exact emulation of the network radio topology and enable support for node mobility. Furthermore, we plan to add a controllable interference and noise injection into the signal path of the emulation hardware.

As chip implementations for IEEE 802.15.4 only provide a specific MAC/PHY configuration in practice, we are also working on the integration of flexible and reconfigurable Software Defined Radio (SDR) modules as IEEE 802.15.4 RIL gateways. This extension opens up interesting opportunities for rapid prototyping of cross-layer communication approaches regarding the PHY in addition.

Finally we will extend our simulation model to support the latest features of the IEEE 802.15.4 specification. In addition, the model must be ported to the current simulation framework version.

ISBN 978-3-903176-13-3

References

[1] K. Wehrle, M. Günes, and J. Gross, "Part III - Higher Layer Modeling," in *Modeling and Tools for Network Simulation*, K. Wehrle, M. Guenes, and J. Gross, Eds., Springer, Mar. 2010, ch. 12.3, pp. 359–485. DOI: 10.1007/978-3-642-12331-3.

[2] M. Alnuaimi, K. Shuaib, and I. Jawhar, "Performance evaluation of IEEE 802.15.4 physical layer using MatLab/simulink," in *2006 Innovations in Information Technology*, IEEE, Nov. 2006. DOI: 10.1109/innovations.2006.301905.

[3] M. Kirsche and M. Schnurbusch, "A new ieee 802.15.4 simulation model for omnet++ / inet," in *Proceedings of the 1st International OMNeT++ Community Summit (OMNeT)*, Sep. 2014.

[4] M. Kirsche, "Selected System Models - IEEE 802.15.4," in *Modeling and Tools for Network Simulation*, K. Wehrle, M. Guenes, and J. Gross, Eds., Springer, Mar. 2010, ch. 12.3, pp. 276–303. DOI: 10.1007/978-3-642-12331-3.

[5] Y. M. Amin and A. T. Abdel-Hamid, "A simulation model of IEEE 802.15.4 GTS mechanism and GTS attacks in OMNeT++ / MiXiM + NETA," *Computer and Information Science*, vol. 11, no. 1, p. 78, Jan. 2018. DOI: 10.5539/cis.v11n1p78.

[6] J. W. Jung and M. A. Ingram, "An rf channel emulator-based testbed for cooperative transmission using wireless sensor devices," in *Proceedings of the Second International Conference on New Technologies, Mobility and Security (NTMS)*, 2008. DOI: 10.1109/NTMS.2008.ECP.22.

[7] J. D. Beshay, K. S. Subramani, N. Mahabeleshwar, *et al.*, "Wireless networking testbed and emulator (winetester)," *Computer Communications*, vol. 73, pp. 99–107, 2016. DOI: 10.1145/2641798.2641809.

[8] G. Z. Papadopoulos, K. Kritsis, A. Gallais, *et al.*, "Performance evaluation methods in ad hoc and wireless sensor networks: A literature study," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 122–128, 2016. DOI: 10.1109/mcom.2016.7378437.

[9] S. Unterschütz, A. Weigel, and V. Turau, "Cross-platform protocol development based on omnet++," in *Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques (SIMUTools)*, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2012, pp. 278–282. DOI: 10.4108/icst.simutools.2012.247711.

[10] M. M. R. Mozumdar, L. Lavagno, L. Vanzago, *et al.*, "Hilac: A framework for hardware in the loop simulation and multi-platform automatic code generation of wsn applications," in *Proceedings of the 9th International Symposium on Industrial Embedded Systems (SIES)*, IEEE, 2010, pp. 88–97. DOI: 10.1109/sies.2010.5551370.

[11] D. S. Buse, M. Schettler, N. Kothe, *et al.*, "Bridging worlds: Integrating hardware-in-the-loop testing with large-scale VANET simulation," in *2018 14th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, IEEE, Feb. 2018. DOI: 10.23919/wons.2018.8311659.

[12] T. Staub, R. Gantenbein, and T. Braun, "Virtualmesh: An emulation framework for wireless mesh and ad hoc networks in omnet++," *Simulation*, 2010. DOI: 10.1177/0037549710373909.

[13] E. Weingärtner, H. vom Lehn, and K. Wehrle, "Device driver-enabled wireless network emulation," in *Proceedings of the 4th International Conference on Simulation Tools and Techniques (SimuTools)*, Barcelona, Spanien: ICST, Mar. 2011. DOI: 10.4108/icst.simutools.2011.245543.

[14] P. Wehner and D. Göhringer, "Internet of things simulation using omnet++ and hardware in the loop," in *Components and Services for IoT Platforms*, Springer, 2017, pp. 77–87. DOI: 10.1007/978-3-319-42304-3_4.

[15] S. Böhm and M. Kirsche, "Looking into hardware-in-the-loop coupling of omnet++ and rosenet," in *Proceedings of the 2nd International OMNeT++ Community Summit (OMNeT 2015), Zurich, Switzerland, September, 2015, CoRR*, 2016.

[16] S. Böhm and M. Kirsche, "Unifying radio-in-the-loop channel emulation and network protocol simulation to improve wireless sensor network evaluation," in *Simulation Science*, M. Baum, G. Brenner, J. Grabowski, *et al.*, Eds., Cham: Springer International Publishing, 2018, pp. 219–238. DOI: 10.1007/978-3-319-96271-9_14.

[17] M. Tüxen, I. Rüngeler, and E. P. Rathgeb, "Interface connecting the inet simulation framework with the real world," in *Proceedings of the 1st International Conference on Simulation Tools and Techniques (SIMUTools)*, Marseille, France: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, 40:1–40:6. DOI: 10.1145/1416222.1416267.

[18] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN)*, Washington, DC, USA: IEEE Computer Society, 2004, pp. 455–462. DOI: 10.1109/LCN.2004.38.

[19] A. Goldsmith, "Wireless communications: Signal propagation and path loss models," Stanford University, California, Scriptum, 2016.

[20] IEEE Standards Association, "Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)," IEEE, IEEE Standards Document - Revision of IEEE Std. 802.15.4™-2006, Sep. 2006, 320 pp. DOI: 10.1109/IEEESTD.2006.232110.