PAPER

Emulation Testbed for IEEE 802.15.4 Networked Systems

Razvan BEURAN[†], **Junya NAKATA**[††], ***Nonmembers***, **Yasuo TAN**[†††], ***Member***, ***and*** **Yoichi SHINODA**[†††], ***Nonmember***

SUMMARY IEEE 802.15.4 based devices are a key component for mobile and pervasive computing. However, their small dimensions and reduced resources, together with the intrinsic properties of wireless communication, make it difficult to evaluate such networked systems through real-world trials. In this paper we present an emulation testbed intended for the evaluation of IEEE 802.15.4 networked systems. The testbed builds on the generic framework of the wireless network testbed QOMB, and adds IEEE 802.15.4 network, processor and sensing emulation functionality. We validated the testbed through a series of experiments carried out both through real-world trials in a smart home environment, and through emulation experiments on our testbed. Our results show that one can accurately, and in real time, execute IEEE 802.15.4 network applications on our testbed in an emulated environment that reproduces closely the real scenario.

key words: *IEEE 802.15.4, wireless network emulation, processor emulation, sensor emulation, network testbed*

1. Introduction

In recent years, the IEEE 802.15.4 standard has been used for a multitude of purposes with application to the areas of wireless personal area networks and home networks. The resulting systems have three characteristics that make it difficult to conduct experiments: (i) they use wireless communication, which is prone to interference and high variability; (ii) they are embedded, hence have small dimensions and few resources; (iii) they are often designed for large-scale deployments.

In this paper we present an alternative to real-world trials for making realistic experiments with IEEE 802.15.4 networked systems by using the emulation paradigm, which provides control, reproducibility, and scalability. Compared to real-world trials, emulation makes it possible for the user to better control the experiment environment, to repeat experiments with ease, and to avoid constraints related to the available number of physical devices or their resource limitations. Compared to simulation, emulation provides increased realism through its most significant feature, which is real-time experiment execution that includes real components. In our particular case this is achieved by running without modification and in real time the emulated device firmware on our testbed.

The emulation testbed that we present here is an extension of the emulation framework provided by the QOMB wireless network emulation testbed [5]. In this paper we present the components used to further extend the functionality of QOMB to make possible experiments with IEEE 802.15.4 networked systems. Such support must take into account two main aspects: (i) emulate the IEEE 802.15.4 wireless communication between the devices; (ii) emulate the functionality of the networked devices themselves, such as firmware execution, and sensing. For functionality emulation, in this work we have focused on JN5139, an 802.15.4 device manufactured by Jennic Ltd. [9].

The novelty of our work consists mainly in the way in which we combined several components, such as network emulation, processor emulation and sensor emulation in order to make possible realistic experiments with IEEE 802.15.4 networked systems. The main contributions are:

1. The design and implementation of a hybrid approach for networked system emulation, that considers independently the PHY and MAC layers of IEEE 802.15.4;
2. A probabilistic model for the IEEE 802.15.4 PHY layer, and an implementation of the IEEE 802.15.4 MAC layer;
3. The emulation of the processor and sensing functionality of a JN5139-based system;
4. A comparison of emulation results obtained on QOMB with those of real-world trials performed in a smart home environment that demonstrate the correct operation of our emulation testbed.

A preliminary version of this work has been previously presented at the AINA 2011 conference [4]. The current paper extends our prior work as follows:

- A more detailed description of the emulation framework that we designed and implemented (see in particular Section 2.3);
- A more thorough presentation of the components related to IEEE 802.15.4 PHY emulation, processor emulation, and IEEE 802.15.4 MAC emulation (Sections 3, 4.1, and 4.2, respectively);
- An extension of the sensing functionality emulation

Manuscript received January 30, 2012.
Manuscript revised April 20, 2012.

[†]R. Beuran is with Hokuriku StarBED Technology Center, National Institute of Information and Communications Technology, Ishikawa, Japan.

[††]J. Nakata is with Uniden Corporation, Tokyo, Japan. This work was done when he was with Hokuriku StarBED Technology Center, National Institute of Information and Communications Technology, Ishikawa, Japan.

[†††]Y. Tan and Y. Shinoda are with Japan Advanced Institute of Science and Technology, Ishikawa, Japan.

DOI: 10.1587/transcom.E95.B.1

component to include humidity and light-level sensors (Section 4.3);
- Two completely new series of experimental results that clearly demonstrate the capabilities and accuracy of the testbed through a comparison with real-world experiments, both in static and mobile scenarios (Section 5).

The remainder of this paper is structured as follows. In Section 2 we present an overview of the emulation testbed. Next, we detail the mechanisms used to emulate the communication of IEEE 802.15.4 networked devices on our testbed (Section 3). Section 4 describes the sub-components required in order to emulate the functionality of the devices themselves. Then we present several experimental results that validate the operation of our testbed (Section 5). Section 6 discusses several research directions that are related to our work. The paper ends with conclusions, acknowledgments, and references.

2. Emulation Testbed Overview

The emulation testbed for IEEE 802.15.4 networked systems that we designed and implemented was created by extending the functionality of the QOMB wireless network emulation testbed. QOMB was initially developed for IEEE 802.11a/b/g network emulation [5].

In this paper we describe the extension of QOMB for IEEE 802.15.4 networked system emulation. This extension presented several technical challenges arising from the fundamental differences between typical IEEE 802.11 and IEEE 802.15.4 systems. Thus, 802.11 systems are usually computers or equivalents, and the applications and protocols are executed on normal CPUs; this makes it easier to emulate them, since it is only requires to model the IEEE 802.11 communication, whereas the applications and protocols are executed as such on the emulation hosts. On the contrary, 802.15.4 systems are typically embedded devices with sensing capabilities and with low computing resources, using particular processors; this leads to the fact that, in addition to modeling the IEEE 802.15.4 communication, it becomes necessary to also emulate the processor and sensing functionality of the target devices. Our approach in this respect is detailed in Section 4. We note that this functionality extension was facilitated by the modular architecture of QOMB and of its components, as demonstrated by our previous work of adding support for active RFID tags [2].

To facilitate understanding our emulation framework, we first introduce its building blocks. The logical hierarchy of the main elements that compose QOMB is shown in Figure 1:

- The hardware infrastructure of the StarBED wired-network testbed;
- The experiment-support software tool named SpringOS, used to manage experiments on StarBED;
- The experiment-support software tool named RUNE, used to control ubiquitous network experiments on StarBED;

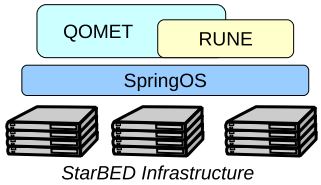


Fig. 1 Logical hierarchy of the QOMB components.

- The wireless network emulation set of tools QOMET.

2.1 StarBED

StarBED is a large-scale wired-network testbed managed by the National Institute of Information and Communications Technology of Japan at the Hokuriku StarBED Technology Center located in Ishikawa prefecture, Japan [13]. StarBED makes available for experiments more than 1100 PCs and the interconnecting network equipment. The large number of experiment hosts, and the versatile network architecture of StarBED make it possible to conduct a wide range of network experiments on this testbed. In our framework, StarBED represents the physical infrastructure of QOMB.

In order to assist users in making large-scale experiments, the StarBED team has developed two software tools, named SpringOS and RUNE.

2.1.1 SpringOS

SpringOS is the fundamental experiment-support software tool for StarBED. SpringOS allows users to perform complex experiments with a large number of hosts in an easy manner [13]. The functions of SpringOS can be divided into two categories:

1. *Experiment preparation*: Configure the experiment hosts and network so that they are ready for being used in experiments;
2. *Experiment execution*: Effectively carry out the experiment by executing in the required order the necessary commands on the experiment hosts.

The use of SpringOS is mandatory for experiment preparation, so as to get access to the StarBED infrastructure. However, the use of SpringOS is optional for experiment execution, and is limited to IP network experiments. In experiments which do no employ IP communication, such as our work related to IEEE 802.15.4, the alternative system RUNE must be used. Hence, in this paper we used SpringOS only for experiment preparation, and we used RUNE instead for experiment execution.

2.1.2 RUNE

RUNE (Real-time Ubiquitous Network Emulation environment) is the experiment-support software tool aimed at making possible on StarBED ubiquitous network emulation experiments with a large number of devices [14]. Differently

from SpringOS, RUNE has no restrictions regarding the type of network used. Thus, RUNE was the ideal choice for driving IEEE 802.15.4 networked system emulation on StarBED.

RUNE experiments are globally managed by a module called *RUNE Master* that is executed on one of the StarBED hosts. RUNE Master communicates with modules called *RUNE Manager*, an instance of which is executed on each StarBED host that takes part in a certain experiment.

The activity of the emulated devices is reproduced by RUNE entities that are called *spaces*. All the spaces that are being executed on an experiment host are controlled by the local RUNE Manager on that host. Spaces communicate with each other via communication channels called *conduits*. Conduits are abstract logical channels for passing messages between spaces with the assistance of RUNE Managers. Network emulation (i.e., recreating realistic communication conditions) is performed by using dedicated spaces, which determine what packets are passed on via conduits and what packets are not (see Section 2.3).

2.2 QOMET

QOMET (Quality Observation and Mobility Experiment Tools) is a set of tools for wireless network emulation that provides the necessary mechanisms for performing this task by reproducing the communication conditions between each and every node in the experiment [3]. QOMET relies on the experiment management mechanisms of StarBED for its distributed execution that results in the emulation of the overall network. The main features of QOMET are:

- *Support for various wireless network technologies*: IEEE 802.11a/b/g, active RFID tag, and IEEE 802.15.4 (presented here);
- *Support for 3D environments*: The topography of the emulated environment (streets and buildings), and the antennas used for communication can be defined both in 2D and 3D;
- *Support for node mobility*: Several models can be used to describe the trajectory of the emulated nodes in the virtual environment (random way, behavioral, etc.).

The core functionality of QOMET is provided through the libraries called `deltaQ` and `chanel`. The `deltaQ` library is used to compute the communication conditions between wireless nodes given a user-defined scenario. The library includes the implementation of models for wireless network technologies, propagation, and mobility. The user-defined scenario represents the input of the `deltaQ` library, and is used to specify the properties of the wireless nodes (position, network technology parameters, mobility patterns), and of the environment in which they are placed (attenuation, shadowing, street and building structures, and so on). These properties are used to create a "virtual world" that corresponds to the emulated scenario, in which the wireless nodes can move and communicate with each other.

The `chanel` library is used to recreate during the live experiment the communication conditions computed by the `deltaQ` library. The `chanel` library is basically in charge of delivering the messages from a wireless node to all the other nodes with which it can communicate according to the scenario; this is done after applying the corresponding network degradation (packet loss, delay, and bandwidth limitation) that was computed by the `deltaQ` library. The `chanel` library is implemented using threads so that the input and output queues can be processed in parallel. One instance of the `chanel` library must be executed for each emulated device.

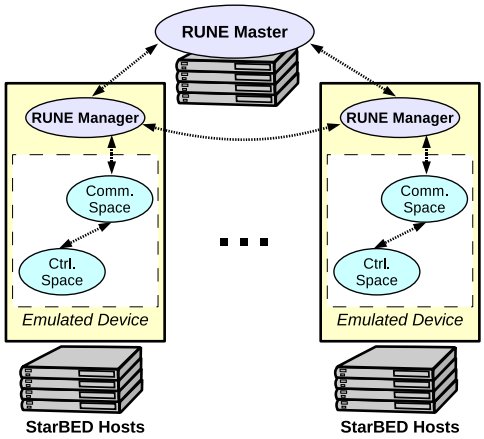


Fig. 2 Architecture of the QOMB emulation framework for IEEE 802.15.4 networked system experiments.

2.3 Emulation Framework

The integration of all the tools that we have mentioned so far effectively gives birth to a new entity, the wireless network emulation testbed QOMB. The architecture of the emulation framework used for our experiments is shown in Figure 2. RUNE is used to manage the experiment that is executed on StarBED hosts via the global RUNE Master and the local RUNE Manager modules. Each RUNE Manager module controls the execution of one or more emulated 802.15.4 devices. SpringOS is only used during the experiment preparation phase, as indicated in Section 2.1.1, and is not mentioned in the figure for clarity reasons.

A particularity of our design is the hybrid approach we take to emulate the 802.15.4 networked devices. We decomposed the activity of the devices in two categories: communication and control. Here communication refers exclusively to the process of sending and receiving data, hence corresponds to the IEEE 802.15.4 PHY layer. The control category assembles several activities related to the functionality of the device itself, such as firmware execution and sensing. Since the JN5139-based devices execute the IEEE 802.15.4 MAC layer on the device processor, we included it in the category of control activities.

The above separation determines the way in which the emulation process takes place in our framework. Thus, each emulated 802.15.4 networked device is made of two elements, which are implemented as RUNE spaces: a *commu-*

nication space and a *control space* (see Figure 2).

2.3.1 RUNE Spaces

The communication space is essentially used to recreate the communication conditions between the emulated 802.15.4 networked devices in the experiment. This is accomplished by employing the `deltaQ` and `chanel` libraries of QOMET. In this context, the `deltaQ` library has been enhanced by adding support for IEEE 802.15.4 PHY layer emulation through the use of probabilistic models (see Section 3).

The control space reproduces the functionality of the emulated devices, in other words their "behavior". The most important component of the control space is represented by the processor emulator, that emulates the execution of the JN5139 device processor, so as to allow using on QOMB the same firmware with that running on actual devices. Other components of the control space are related to IEEE 802.15.4 MAC emulation, as well as to sensing functionality emulation (see Section 4).

2.3.2 Jennic JN5139

As mentioned already, in this work we focused on the device named JN5139, which is manufactured by Jennic Ltd. (now part of NXP Semiconductors). The JN5139 is a general-purpose micro-controller that integrates a 32-bit RISC processor with a fully compliant 2.4 GHz IEEE 802.15.4 transceiver. The processor has an OpenRISC architecture and operates at 16 MHz. The memory consists of 192 kB ROM for system code, including protocol stack, and 96 kB RAM for system data and optional program code.

The JN5139 has other features such as comparators, timers, counters, and various kinds of interfaces. In particular, the device that we emulate has on-board temperature, humidity and light-level sensors, as well as an LCD display.

3. Communication Emulation

The communication space in Figure 2 is in charge of reproducing the communication conditions between the emulated 802.15.4 devices. Practically, the communication space employs the `chanel` library to accomplish this task, hence to deliver or drop the frames received from the control space, depending on the state of the network.

The communication conditions that are recreated by the `chanel` library are provided by the `deltaQ` library, which was enhanced to include support for IEEE 802.15.4 PHY layer emulation. The modifications refer to the use of specific models for the different characteristics of the 802.15.4 PHY layer, as well as for electromagnetic wave propagation.

3.1 IEEE 802.15.4 PHY Emulation

In order to emulate the IEEE 802.15.4 PHY layer, a series of models must be used to compute the frame error rate, delay and bandwidth limitations that characterize the communication conditions of this layer. These parameters are then applied to the frames that are received by the communication space from the control space before sending them to the other emulated devices in order to recreate the communication conditions of the user-defined scenario.

The computation is done by starting from the properties of the virtual world in which the emulated devices are located, and by taking into account the properties of the 802.15.4 PHY layer, in a hierarchical series of models.

For electromagnetic wave propagation we use the log-distance path loss model [17], which gives the received power at a distance d, P_r, as function of the received power at the distance of 1 m when transmitting at 1 dBm, P_{r0}, the attenuation coefficient α, the wall attenuation W, and the standard deviation σ of the shadowing component (which has a normal distribution with mean 0):

$$P_r = P_{r0} - 10\alpha \cdot log_{10}(d) - W + X_\sigma. \quad (1)$$

The frame error rate FER_S is computed in a probabilistic manner from the received power:

$$FER_S = FER_{S0} \cdot e^{S-(Pr-N)-N_{th}}, \quad (2)$$

where S is the receive sensitivity threshold of the device transceiver (provided by the manufacturer), FER_{S0} is the frame error rate when P_r reaches the threshold S (specified by the IEEE 802.15.4 standard), N is the background noise in the virtual environment, and N_{th} is the thermal noise.

Note that FER_{S0} is specified by transceiver manufacturers for a particular frame size, that we denote by FS_S, and which is equal to 20 bytes for IEEE 802.15.4. Therefore, the result obtained from Equation (2) is specific to this frame size. To extend the calculation of the frame error rate to frames with arbitrary sizes, we use:

$$FER = 1 - (1 - FER_S)^{FS/FS_S}, \quad (3)$$

where FS denotes the size in bytes of the frame for which the generic frame error rate FER is calculated.

At PHY layer, frame delay is given by transmission delay and propagation delay. Since the propagation delay, i.e., the time needed for electromagnetic waves to travel from sender to receiver, is very small for 802.15.4 networks (under $1\mu s$), we only consider the transmission delay, D:

$$D = T_{PHY} + T_{PSDU} + T_{IFS}, \quad (4)$$

where T_{PHY} is the duration of the PHY header, T_{PSDU} is the duration of the PHY payload, and T_{IFS} is the duration of the inter-frame spacing. T_{PHY} is constant, and T_{IFS} values depend on frame size, being 192 μs for frames under 18 bytes, and 640 μs for larger frames. T_{PSDU} is computed as:

$$T_{PSDU} = (8 \cdot F_{PSDU})/R, \quad (5)$$

where F_{PSDU} is the size of the PHY payload in bytes, and R is the operating rate.

Considering the previous results, the amount of effective bandwidth that is available at the PHY layer is:

Table 1 Parameter values for IEEE 802.15.4 PHY emulation

Parameter	Value
Attenuation coefficient, α	4.02
Shadowing parameter, σ	0.0
Receive sensitivity, S	-96 dBm
Error rate threshold at S, FER_S	0.01
Frame size at S, FS_S	20 bytes
PHY header duration, T_{PHY}	192 μs
Inter-frame spacing, T_{IFS}	192 or 640 μs
Thermal noise, N_{th}	-105 dBm
Operating rate, R	250 kb/s

$$B = \frac{T_{PHY} + T_{PSDU}}{D} \cdot R. \quad (6)$$

3.2 Practical issues

The parameter values for the 802.15.4 PHY layer model equations presented here are summarized in Table 1. These values were used when performing the experiments that will be described in Section 5.

The computed frame error rate, delay, and bandwidth (FER, D, and B, given by Equations 3, 4, and 6, respectively) represent the IEEE 802.15.4 PHY communication conditions that correspond to the user-defined scenario. In practice, the channel library used by the communication space in our implementation only uses the frame error rate parameter. This is because the delay and bandwidth in our implementation are correctly reproduced as an effect of the operation of the 802.15.4 MAC emulator in the control space. Thus, the MAC emulator will output data with the correct delay and bandwidth characteristics of the 802.15.4 MAC layer (including frame retransmission).

Regarding frame error rate emulation, the following decisions are made in the communication space for each frame incoming from the control space:

- If $FER = 0$, the frame is passed on as it is;
- If $0 < FER < 1$, an error is introduced in the frame with a probability equal to FER, and the frame is then passed on;
- If $FER = 1$, the frame is dropped.

4. Functionality Emulation

The emulation of the functionality of the JN5139 device itself is performed by the control space that was shown in Figure 2. The following three features were required:

- *Processor emulation*: Emulate the execution of the OpenRISC processor of JN5139;
- *IEEE 802.15.4 MAC emulation*: Emulate the 802.15.4 MAC protocol;
- *Sensing functionality emulation*: Emulate the functionality of the JN5139 sensors.

The relationship between the three components that implement these features in our emulation framework is shown in Figure 3. The JN5139 firmware is executed by the OpenRISC processor emulator exactly as it would be executed on the real device. The firmware also receives input from the sensor emulator, which is sensing the emulated virtual environment, similarly to how the sensors would be used in the real world.

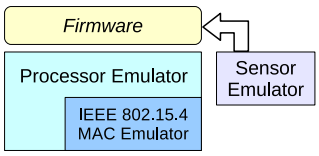


Fig. 3 Relationship of the control space components.

The most important difference between the emulation framework and the actual devices concerns the 802.15.4 MAC emulator. In the actual devices, the 802.15.4 MAC protocol is implemented as a proprietary library by the manufacturer of JN5139, Jennic Ltd. As such, it is executed on the OpenRISC processor, similarly to the device firmware. However, in our framework the processor emulator calls as needed an equivalent 802.15.4 MAC implementation that is executed natively on the PC host on which the emulated device is run. The rationale behind this decision and its implications will be discussed in Section 4.2.

Amongst the three components discussed here, processor emulation is the most device specific. Nevertheless, our implementation could be employed for devices other than JN5139 provided that they use the same OpenRISC processor (or a compatible one). The MAC emulation implementation is based on the IEEE 802.15.4 standard [8], hence it is device independent. However, the source code was designed and compiled for the particular operating system of the PC hosts used in our experiments, which is FreeBSD. The models used for sensor emulation are relatively generic as well, hence usable in other cases too. Still, the particular values of the parameters we used during our experiments were obtained through calibration using the real JN5139 devices that we employed, and may need to be adjusted before being used in the emulation of other devices.

4.1 Processor Emulation

The ability to execute the same firmware with the real devices is in our view one of the most important requirements for enabling realistic emulation experiments with 802.15.4 networked systems. The processor emulator that we implemented is named ORE (OpenRISC Emulator), and it supports the hardware components of the JN5139 microcontroller, including memory and counters. Therefore ORE can execute the same firmware with the JN5139 devices.

ORE intercepts the function calls related to the 802.15.4 MAC protocol, and passes them to the 802.15.4 MAC emulator module (see Section 4.2) without directly executing any code. In the case the function calls produce an output, this output is passed on to the firmware upon being received from the MAC emulator. This is the most significant difference between firmware execution by the processor

emulator, and that taking place in the actual devices.

Thus, in our implementation, the ORE processor emulator handles all firmware instructions *except* those related to 802.15.4 MAC operation, which are passed on to the MAC emulator. The relationship of these two components of the control space was shown in Figure 3, which represents the MAC emulator as a distinct component which is nevertheless tightly integrated with the processor emulator.

4.2 IEEE 802.15.4 MAC Emulation

The IEEE 802.15.4 MAC functionality in our emulation framework was recreated by implementing an equivalent to the proprietary Jennic library that is used on the JN5139 devices. This alternative 802.15.4 MAC implementation, named JLE (Jennic Library Emulator), runs as a separate module in the control space, and not on the processor emulator: a difference compared to the actual device. The reason for introducing this difference in our design is the following. Supporting 802.15.4 MAC protocol functions would require to also implement the 802.15.4 transceiver. Moreover, this hypothetical solution requires executing the MAC implementation on the emulated device processor.

In our implementation, the equivalent functionality is provided by JLE, which is executed *natively* on the CPU of the experiment host. Native execution leads to a significantly faster execution compared to the case when the instructions would be executed by the processor emulator. For instance, when considering an experiment host with a 1.6 GHz CPU versus the 16 MHz RISC processor of JN5139, the execution could be up to about 100 times faster. Hence, our approach has the advantage of simplifying the overall emulator design, and also of speeding up execution. As a result, emulation performance is significantly improved, and our emulation testbed can support more easily the live execution of 802.15.4 application firmware.

A potential issue in this context is the following. Given the faster execution of the MAC primitives by the MAC emulator, one may think that the 802.15.4 MAC layer will actually run faster on our emulation testbed that it would do on real JN5139 devices. We stress that the execution speed difference is compensated in regard to overall timing by the absolute delays built into the MAC protocol, which correspond to absolute delays that are reproduced on our emulation testbed. This makes that, although the execution of the MAC primitives is faster (freeing CPU time for other tasks), overall the MAC emulator produces and consumes messages in the same way with the actual devices, and in synchronization with the wall clock.

Nevertheless, the execution of the firmware instructions on our emulation testbed is done in a loosely-coupled manner compared to that in the actual devices. This is because on the emulation testbed general-purpose and MAC-related instructions are executed by two different entities, ORE and JLE, whereas they are executed sequentially by the OpenRISC processor in the actual devices. The sequential nature of these operations is enforced in our implementation, and we have detected no out-of-order issues in our experiments. Moreover, we have not noticed any adverse effect of this loose synchronization on the operation of the emulated devices, hence it appears to pose no problem on our emulation testbed.

As a related matter, we note that we use no specific distributed time synchronization mechanism in our testbed. Instead, each time-critical component synchronizes itself with the local clock of the PC on which it is run by timing its own execution. Moreover, all the PCs are synchronized with a time source via NTP (Network Time Protocol). This method ensured sufficient time accuracy for our purposes.

4.3 Sensing Functionality Emulation

The JN5139 device has on-board temperature, humidity, and light-level sensors. All the three sensors are supported by the processor emulator from the point of view of the interaction with them through a memory-mapped register access mechanism. Nevertheless, the sensor data obtained from these registers must be close to reality in order to make realistic experiments possible. Thus, it was necessary to also model the sensing functionality of the sensors itself, and this was done separately for each type of sensor.

4.3.1 Sensor Models

One of the widely-used models for taking into account the dynamic behavior of sensors is the "thermal time constant model", often used for thermal sensors [12]. Following this model, the equation for computing the temperature reading of the thermal sensor at time t after the sensor is brought into a certain environment, that we denote by $T(t)$, is:

$$T(t) = T_A - (T_A - T_0)e^{-t/C_{th}}, \tag{7}$$

where T_A is the ambient environment temperature to which the sensor is adapting, T_0 is the initial temperature reading of the sensor upon entering the environment, and C_{th} is the thermal time constant of the sensor.

Intuitively, the time constant C_{th} indicates how quickly the reading of the sensor approaches in an asymptotic manner the ambient temperature. A low value of C_{th} leads to a faster adaptation, and a large value leads to a slower one.

We consider the dynamic behavior of the humidity sensor to be similar to that of the thermal sensor, as it has to adapt its reading to the ambient humidity in a resembling manner. Therefore, we use a similar equation for modeling the humidity reading of the sensor at time t after the sensor is brought into a certain environment, denoted by $H(t)$:

$$H(t) = H_A - (H_A - H_0)e^{-t/C_{hum}}, \tag{8}$$

where H_A is the ambient environment humidity to which the sensor is adapting, H_0 is the initial humidity reading of the sensor upon entering the environment, and C_{hum} is the humidity time constant of the sensor.

For practical purposes it can be considered that the

reading of a light-level sensor reflects instantaneously the illumination of the environment into which it is brought. Therefore, the light-level reading of the sensor at time t after the sensor is brought into a certain environment, that we denote by $L(t)$, is given in our emulation framework by:

$$L(t) = L_A, \tag{9}$$

where L_A is the ambient environment light level to which the sensor is adapting. We note that the above equation is a particular case of the previous ones where the time constant is made equal to 0, signifying infinitely fast adaptation.

4.3.2 Ambient Parameters

The above models were implemented as a third component of the control space, in charge of sensor emulation (cf. Figure 3), and used to compute the sensor readings that are provided to the firmware executed in the emulation framework. The input for the sensor models consists of ambient characteristics (temperature, humidity, and light level) for each of the areas in the emulated virtual environment.

Regarding ambient parameters, there are two important issues that need further clarification: (i) how the ambient parameters are determined for a certain target environment; (ii) how the ambient parameters are managed within the emulated environment.

The ambient parameters for the target environment must be determined from sensor readings obtained from sensors placed in that environment. In the most basic scenario, one sensor is used for each area of interest, as we have done for the experiments reported in this paper. In particular, we used for this purpose the sensor readings of the sensors built into JN5139 devices that were deployed in the smart-home environment used in our real-world experiments. In order to gather more detailed information, it is also possible to place multiple sensors for each area, for instance 8 sensors per room: four in each corner of the floor, and four in each corner of the ceiling. Moreover, another possibility is to use a three-dimensional grid of sensors deployed for each area of interest for which ambient characteristics need to be determined in great detail.

Although in this paper we have only used the first approach (one sensor per area), the other two approaches were used in experiments that we have carried out for different purposes. Advantages of the first method are its simplicity, and the fact that the average sensor readings of the real sensor in a certain interval can be passed directly as input ambient parameters to the emulated sensor (assuming that their spatial position is equivalent, as it was in our experiments). If the position of the emulated sensor is different from those of the real ones, then the multiple sensor readings obtained in the other approaches should be combined in order to estimate the ambient parameters at the location of the emulated sensor. For temperature, for instance, we did this using heat propagation equations.

As for the management of ambient parameters in the emulated environment, we note that the sensor value corresponding to a certain emulated device takes into account the area in which that device is located. Thus, static nodes are assigned a fixed position during the entire duration of the experiment, and this position is used to determine the area in which they are located. Mobile nodes on the other hand may be located in different areas depending on the current time in the experiment. The area in which a mobile node is located at a certain time is determined by taking into account the trajectory of the node and the topography of the virtual environment. The area information is then used to calculate the corresponding ambient parameters for that moment of time and place based on the procedure explained in the previous paragraphs.

It is important to note that our current approach assumes that ambient conditions are constant, and this is a good approximation for our experiments, which never exceeded 15 minutes. However, for longer experiments it is necessary to also take into account the variation of the ambient conditions, and their dependence on time.

Moreover, we emphasize the fact that the time constant sensor characteristics that we used as parameters in our models are not provided by all manufacturers. In that case, either generic values can be used, or the characteristics of the sensors can be determined by the researchers themselves through simple field trials; the latter is the approach that we have taken in our experiments. Table 2 shows the values that we determined for the sensors in the JN5139 device by calibrating the time constant parameters through real-world measurements.

Table 2 Empirically-determined time constants for the sensor emulation models

Constant name	Empirical value
Thermal time constant, C_{th}	340
Humidity time constant, C_{hum}	20

5. Experimental Results

In this section we present several series of experimental results in static and mobile scenarios that validate the functionality of our testbed and demonstrate its applicability.

5.1 Experiment Methodology

The methodology used for our testbed validation is to perform similar experiments by using actual IEEE 802.15.4 sensor nodes in a real environment, and also by using the emulation testbed that reproduces an equivalent environment.

5.1.1 Sensor Nodes

The actual IEEE 802.15.4 sensor nodes used in our experiments are based on JN5139. The application firmware used was provided by the company Fujitsu Nagano Systems Engineering Ltd., which also manufactured the sensor nodes

that we effectively used. The functionality of the sensor nodes is divided into two classes, based on their role from an 802.15.4 network point of view, namely:

- *Coordinators:* Form the root of the 802.15.4 network star topology, and can bridge to other networks if configured to do so;
- *End devices:* Connect to the root node of the star topology, and perform the sensing function.

As a consequence, any experiment must include at least one coordinator node, and one or more end device nodes. While the coordinator only has network-related functionality, the end devices are also performing the sensing activity. In particular, each end device, once connected to a coordinator, will send at regular intervals (configured to 10 s) sensor data to the coordinator.

In our experiments, the coordinators were connected via a USB cable to a notebook PC. The USB cable was used in our case only to power the devices, but could also be used to log data from coordinators. On the other hand, the end devices were all battery powered. Note that we didn't consider the effects of battery depletion on experimental results, since all our experiments had a relatively short duration. This issue is being considered as future work.

In order to capture the traffic in the 802.15.4 network, we employed the CC2531 802.15.4 USB dongles from Texas Instruments [19]. The associated software was used to log all the communication between the nodes in our experiment onto the notebook to which a certain dongle was attached. The dongles were placed near the coordinator nodes in our experiment, such as to have a "view" of the network similar to that of a coordinator, but without the additional overhead of using the coordinator itself for this task. This approach also relied on the fact that the USB dongles and JN5139 devices have the same receive sensitivity.

5.1.2 iHouse Smart Home

The devices that we emulated, Jennic JN5139, are well suited for being used in a smart home environment, given that fact that they are equipped with sensors for temperature, humidity and light level. As a consequence, we performed real-world trials in such a smart home environment, located in the vicinity of our research center, in Ishikawa prefecture, Japan. A photograph of the smart home, named *iHouse*, is shown in Figure 4. The room layout for each floor of iHouse is presented in Figure 5.

The iHouse includes networked equipment such as air conditioners, windows, blinds, etc. It is possible both to obtain the state of the equipment (e.g., the temperature reading and air flow speed for air conditioners, the state of the windows, or the deployed length for the blinds) and to control the equipment (e.g., change the temperature setting of the air conditioner, open or close the windows, extend or retract the blinds) using the network connectivity of these appliances. A detailed description of iHouse in available in [18] (in Japanese).

Fig. 4 The smart home iHouse used in our real-world trials and emulation experiments.

5.1.3 Basic Emulation Environment

Our emulation environment was based on the room layout of iHouse shown in Figure 5. This allowed us, first of all, to illustrate the capabilities of our 802.15.4 network emulation testbed in a home network scenario. Secondly, this made it possible to use the actual house as a validation environment for our emulation testbed. This is important considering that the emulation testbed is intended for extending the scale of the experiments that can be performed in the actual house. For instance, one could use more sensors in the virtual environment than physically available in the real house, or do experiments in different environment conditions (temperature, humidity, etc.) than it would be possible to do in the real world at a certain time.

The steps necessary in order to emulate the iHouse on our testbed are summarized below. Note that although these steps are given for the particular case of iHouse, the methodology is by no means restricted to this environment, and could be in principle applied to any other environment for which a virtual representation needs to be created. The required actions for this purpose are:

1. *Model the topography of the environment:* We created a simplified 3D representation of iHouse in QOMET based on the construction data used to build the actual house. Simplifications were done in the shape of the rooms and of the stairs, and also by eliminating components that are not relevant for our experiments, such as the roof;
2. *Determine the communication properties of the environment:* We conducted a series of basic measurements in which we placed two sensor nodes at a known distance with respect to each other in three scenarios: (i) unobstructed communication; (ii) communication through a wall of iHouse; (iii) communication through the ceiling between the first and the second floors of iHouse.

For the basic experiments mentioned above, the JN5139 nodes were located at 2 m from each other in all three scenarios; the experiments were repeated 5 times. By comparing the signal strength results, more specifically the RSSI (Received Signal Strength Indication) values, for the

Table 3 Empirically-determined communication parameters for iHouse

Parameter name	Empirical value
Wall attenuation	6.03 dBm
Ceiling attenuation	14.30 dBm
Indoor boost	8.38 dBm

unobstructed communication case with the other two cases, we were able to estimate the wall attenuation, as well as the ceiling attenuation for iHouse. We also conducted a similar unobstructed experiment in an outdoor setting. Comparing the RSSI values for the indoor and outdoor settings, we were also able to determine the boost of the signal strength in the indoor case, typically caused by reflections from the walls. These empirical values are given in Table 3.

By using the communication parameter values in the QOMET scenario representation, we were able to create a virtual environment that has similar communication properties with iHouse. This environment was used as a basis for all our subsequent emulation experiments. Note that, in addition to obstacle attenuation, QOMET also uses a propagation attenuation coefficient and a shadowing constant in the communication model (see Equation 1). For the propagation attenuation coefficient we used the value 4.02, which is considered typical for indoor environments [7]. We also made the simplifying assumption of having no shadowing effects, hence the σ parameter in Equation 1 was 0.

5.2 Static Experiments

Our real-world trials in iHouse started with several series of static experiments using up to 2 coordinators and 7 end devices. Their placement is shown in Figure 5, where coordinators are denoted by "CN", and end devices by "ED".

For the first series of experiments, only the coordinator at the second floor was activated, and all the end devices had to associate with it. In the beginning, only one end device was activated at a time for a period of 10 minutes. The purpose of these preliminary experiments was to validate whether each end device can communicate with the coordinator. The results showed that this was indeed the case. We also determined that frame error rate had acceptable values for most rooms (not exceeding 13% in the worst case), and that the variability of RSSI was low (worst-case standard deviation not exceeding 3).

We then proceeded with more realistic experiments, in which again the coordinator at the second floor was used, and all the 7 end devices were active during the entire experiment, which lasted 10 minutes. This type of experiment was repeated 2 times. For each experiment we determined the average RSSI, the frame error rate, as well as the average temperature, humidity and light-level readings.

The same experiment was performed through emulation on QOMB. The virtual environment was based on that described in Section 5.1.3; the coordinator and end devices were also placed in the virtual environment at equivalent locations with those in iHouse. The average RSSI values computed by QOMET were used to calibrate the environment attenuation parameters so as to account for the complexity of the actual 3D structure of iHouse. The emulation experiments were then run in a similar manner to the previously-described real-world trials.

In Figure 6 we compare the average RSSI per end device in the two real-world trials with the RSSI value computed by QOMET. Each end device is labeled with the name of the room in which it was placed. The experimental values match well our computation, with the largest differences, of about 10%, being observed for the "Dining" room.

In Figure 7 we compare the frame error rate for each end device in the two real-world trials with the FER computed by QOMET, and also with the FER measured for the emulation experiment on QOMB. Again the real-world trial results match well both the computation and the measurements done on the emulation testbed. We noted a larger difference for the "Kitchen" area. However, this difference of less than 15% is only between the measured FER and the other results, and we expect that it is caused by the relatively low number of packets involved in the experiment. Sending one sensor data packet every 10 s resulted in a total of 60 packets during our 10 minute experiments; this relatively small number of packets can lead to statistical differences when packet loss is introduced in a probabilistic manner as done in our implementation. We also noticed an unexpectedly high FER value for the "Bedroom" area in the first real-world trial, that we attribute to transient effects, since the other trials produced values closer to our expectations.

In Figures 8 and 9 we compare the results obtained for temperature and light level, respectively (humidity results were similar, hence we omit them because of space considerations). The virtual environment settings were configured to match the conditions in iHouse, and our results show that the sensor readings of the emulated devices match well those of the actual ones. We note that the light level for the "Bedroom" area differs by about 15% between the two real-world trials; we assume this difference was caused by a slight change of the orientation of the actual sensor in-between experiments, hence it is not significant.

For another series of experiments, we activated both coordinators shown in Figure 5. In this case, the end devices at the first floor of iHouse associated with the coordinator at the first floor, and the end devices at the second floor of iHouse with the coordinator at the second floor. The separation was ensured by using different identifiers for the two networks. Similarly to the results presented before, a good match was observed between real-world trials and emulation; results are omitted because of space considerations.

5.3 Mobility Experiments

The experiments presented so far did not assess the dynamic behavior of the sensors that we emulate, since they only focused on static conditions. In order to investigate this second aspect, we performed several mobility experiments both in iHouse and through emulation.

We present here a full-scale experiment in which a per-

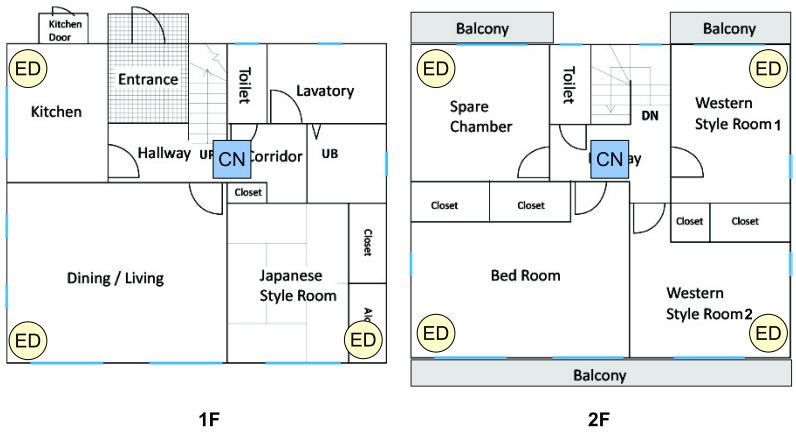


Fig. 5 Setup for static experiments indicating the position of the sensor nodes in iHouse.

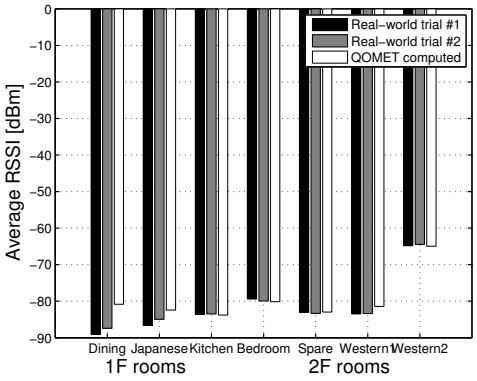


Fig. 6 Average RSSI in the real-world trials, and as computed by QOMET for each room in iHouse (static experiments).

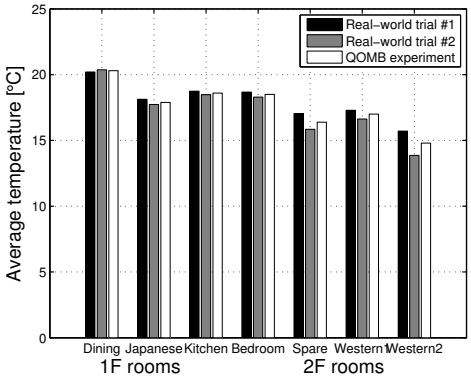


Fig. 8 Average temperature in the real-world trials, and as measured on QOMB for each room in iHouse (static experiments).

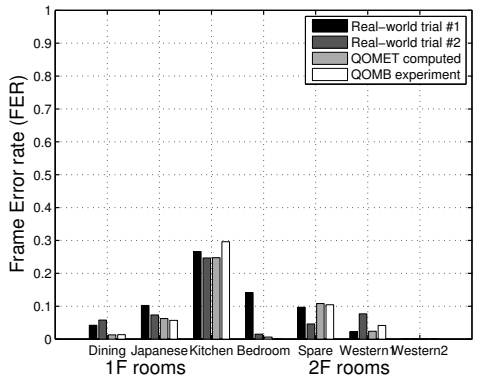


Fig. 7 Frame error rate in the real-world trials, as computed by QOMET, and as measured on QOMB for each room in iHouse (static experiments).

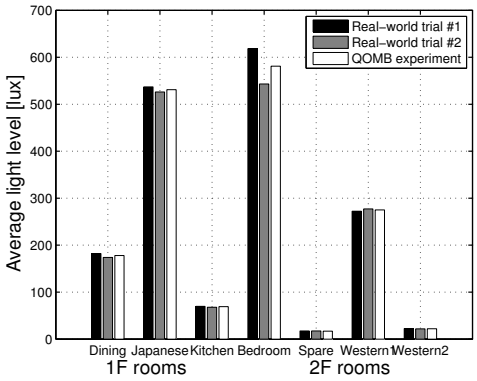


Fig. 9 Average light level in the real-world trials, and as measured on QOMB for each room in iHouse (static experiments).

son moves through all the rooms of iHouse while carrying a sensor node working in end-device operation mode. This experiment lasted a total of 15 minutes, with about 2 minutes spent in each of the seven rooms of iHouse. The experiment was repeated two times, trying to ensure that the motion followed the same trajectory. The motion path is shown in Figure 10. The person moved from the entrance area at the first floor (the start position at time 0 minutes) to

Table 4 Motion start time for each room in iHouse

Area name	Start time [min.]	Area name	Start time [min.]
Entrance	0	Western 1	9
Kitchen	2	Western 2	11
Dining	4	Bedroom	13
Japanese	6	Spare	N/A

the kitchen, then to the dining and Japanese rooms. Following that, the person climbed to the second floor, where he entered the Western room 1, the Western room 2, then the bedroom. Finally the person entered the spare room, where he stayed until the end of the experiment, at time 15 minutes. Motion start times in minutes are shown in Table 4.

Figures 11 and 12 show the variation of the temperature and humidity versus time, respectively. The comparison is done between two real-world trials and the two corresponding emulation experiments. Although the result variation is similar in the two real-world trials, the absolute values are different, reflecting slightly different conditions in the areas of iHouse at different moments of time (the two real-world trials were done at half an hour interval). As a consequence, when running the equivalent emulation experiments we used different settings for the temperature and humidity of the various areas of interest in the virtual environment.

In the case of temperature, results match very well, with the most important differences observed in the interval around the time 7 minutes in Figure 11. The explanation is that this interval includes the person's motion on the stairs. The stairs were not modeled from a sensing point of view in our emulation environment, i.e., they were not assigned specific characteristics, such as temperature and humidity. In the actual iHouse, the temperature on the stairs was lower than that of the other areas, and this lead to lower sensor readings while the person used the stairs. Despite this, the maximum local difference between real-world trial and emulation temperature measurements in our experiments is of about 0.5°C (i.e., about 2.5% of the average temperature).

For humidity, the match is less good, and this is mostly because the humidity sensor of the actual devices does not have a very good resolution. Our experimental data shows that this resolution is of about 0.5% (see, for instance, the oscillations in the interval between 1 and 4 minutes in Figure 12 for the humidity sensor readings obtained in real-world trial #1; by contrast, the temperature sensor has a much finer resolution, estimated to be around 1/100 °C, as demonstrated by the smooth variation in Figure 11). The sensor calculations in our emulator uses floating point arithmetic in all cases, which leads to some intrinsic differences between real-world trial and emulation humidity sensor readings. One other thing to note in Figure 12 is the difference in the interval around the time 7 minutes, which is caused by the motion on the stairs in iHouse (which had a higher humidity compared to other areas). The maximum local difference between real-world trial and emulation humidity measurements in our experiments is of about 1.5% (i.e., about 3.5% of the average humidity value). The explanation we have given in the previous paragraph regarding temperature differences applies in this case as well.

Table 5 Mean differences between real-world trial and emulation sensor data (mobility experiments)

Sensor data type	MAE	RMSE
Temperature (experiment #1)	0.09°C	0.13°C
Temperature (experiment #2)	0.08°C	0.10°C
Humidity (experiment #1)	0.41%	0.53%
Humidity (experiment #2)	0.30%	0.42%

The differences between our results from real-world trials and emulation experiments are quantified in a global manner in Table 5 by using the mean absolute error (MAE) and the root mean square error (RMSE) as metrics. The results show that the global error is around 0.1°C for temperature, and around 0.5% for humidity, which are equivalent to about 0.5% and 1% of the average values, respectively.

We note that the light-level sensor data results for our mobility experiments were inconclusive because the sensor was occasionally covered during the movement. Moreover, given the instant variation of the readings for the light-level sensor, it would be difficult to provide through emulation dynamic results that are close-enough to real-world trials, unless the virtual environment model of iHouse includes all the light sources (windows, lights, etc.), their position, luminosity, etc. Therefore, in this work we only relied on static scenarios for light-level sensor emulation evaluation, as presented in Section 5.2.

5.4 Discussion

Our experimental results demonstrated that QOMB is able to emulate with sufficient accuracy various static and mobile scenarios in our smart home environment. These experiments allowed us to validate the operation of the emulation testbed, and to prove that it performs according to expectations when compared to real-world trials. Our validation procedure was twofold:

- From a network perspective, through results such as average RSSI and frame error rate;
- From a sensing perspective, through results such as average temperature, humidity, and light level.

In all cases we observed a good match between the real-world trial measurements and those made on QOMB for emulation experiments, both for network parameters and sensor readings. Our experiments demonstrate that the sensing application behaves in the same manner both when executed on our 802.15.4 emulation testbed and in the real-world trials.

Such realistic experiments make possible several ways to employ our emulation testbed, as follows:

1. *Use it in advance of real-world trials*: To ensure that a certain IEEE 802.15.4 network application functions as expected in different operation scenarios;
2. *Use it in addition to real-world trials*: To enable experiments that require repeatability and control of the conditions, or to perform larger-scale experiments;

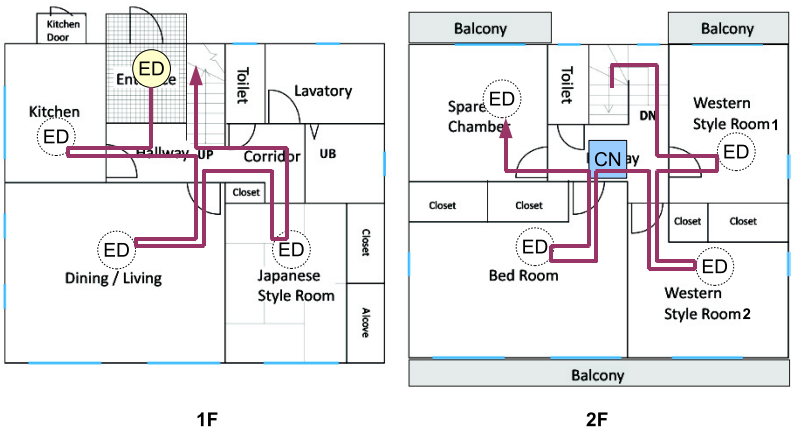


Fig. 10 Setup for mobility experiments in iHouse showing the trajectory of the mobile sensor node.

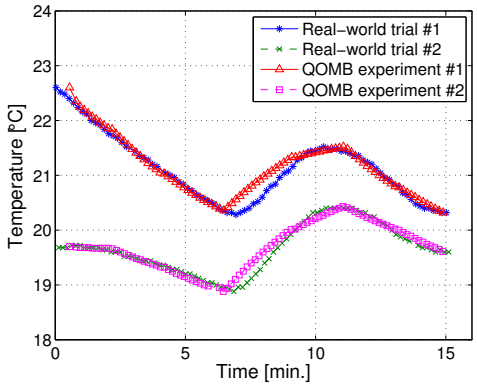


Fig. 11 Temperature variation in the real-world trials, and as measured on QOMB (mobility experiments).

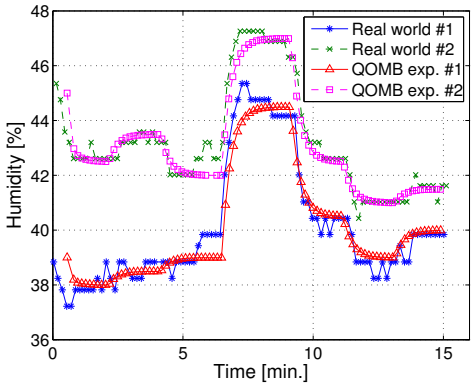


Fig. 12 Humidity variation in the real-world trials, and as measured on QOMB (mobility experiments).

3. *Use it as a component of real-world trials*: Integrate real devices with emulated ones in the same experiment so as to combine the benefits of the two techniques.

While all the uses above are of great importance, we believe that the third approach of integrating real and emulated devices is particularly promising. The main advantage of such an approach is the ability to make experiments with devices that are too difficult or too expensive to model, without requiring to actually deploy the full system, and also while maintaining the benefits of emulation (repeatability, control, scalability). In the smart home context, we suggest that our testbed could be used for the development of an intelligent environment control system that uses the data provided by the sensors. Instead of modeling various actuators, such as air conditioner and blinds, one could use the actual devices in iHouse, whereas the sensing aspects would be emulated on QOMB. In this manner one could validate the correct operation of the system both from the point of view of the communication protocol, and also by making experiments in a wide-range of scenarios.

6. Related Work

To the best of our knowledge, no framework having an identical functionality with our IEEE 802.15.4 networked system emulation testbed currently exists. There are, however, several approaches and tools that are related to our research. We present next the most important of them.

The system that is closest to our design is perhaps the COOJA simulator for the Contiki OS [15]. COOJA uses an original cross-layer simulation approach to execute Contiki OS programs either as compiled native code on the host CPU, or in an instruction-level device CPU emulator (e.g., TI MSP430). Similar to COOJA, TOSSIM is a TinyOS mote simulator targeting the development of sensor network applications [11]. It is said to scales to thousands of nodes, and it compiles directly from TinyOS code. As with our testbed, developers can test their algorithm implementations, albeit through simulation. Another system, ATEMU, is a simulator for systems based on the Atmel AVR processor [16]. It also includes support for other peripheral devices on the

MICA2 sensor node platform [6], such as the radio interface. ATEMU can be used to conduct studies in a controlled simulation environment, and is compatible at binary level with the MICA2 hardware.

All these frameworks are essentially simulators focusing on particular processors and the code execution for those processors. Therefore, they often use very simple models for the wireless communication (for instance, ATEMU only supports free-space propagation) and for sensing. Moreover, they provide no guarantee as to how much slower or faster than the wall clock experiment execution is. Our testbed has none of these drawbacks, as we use a more advanced probabilistic model for wireless communication, we included models and emulation of the sensing functionality, and we execute the firmware in real time.

SensorRAUM is another related project whose goal is to create a quasi-realistic virtual representation of the physical environment [1]. This representation makes it possible to have interactions between real sensors and the virtual world. By emulating the sensors too, we give the user more control over the experiment.

The general-purpose testbed that is Emulab includes a sensor network testbed. The sensor network testbed consists of 25 MICA2 motes that can be used remotely for experiments [20]. Another sensor testbed is MoteLab [21], which consists of a set of permanently deployed sensor network nodes connected to a central server which handles their management. Even though both Emulab and MoteLab are controlled environments, the devices used in these testbeds are real, hence subject to potential wireless interferences. Moreover, no mobility experiments are possible on these testbeds. Mobile Emulab uses robots to perform motion in a reproducible manner [10]. However, the communication is still subject to potential undesired influences. In addition, the mobility range and speed of the robots are limited.

Several wireless network emulation testbeds exist, but they are mainly related to IEEE 802.11 networks. A system such as TWINE uses computer models to perform real-time experiments [22]. Thus, it avoids undesired interferences and side effects, in a similar manner to the approach used by QOMB for IEEE 802.15.4. However, our testbed implements both 802.15.4 network emulation, and the emulation of the hardware of actual 802.15.4 devices, resulting in a more realistic system.

7. Conclusion

In this paper we presented an emulation testbed for IEEE 802.15.4 networked systems that can be used to perform repeatable experiments in a controlled environment. The testbed is built by extending the functionality of the wireless network emulation testbed called QOMB, that was previously created by the integration of the large-scale wired-network testbed StarBED with the wireless network emulation set of tools QOMET.

The extension of QOMB for 802.15.4 experiments was accomplished by adding, first of all, support for the 802.15.4 PHY layer, through a probabilistic communication model, and for the 802.15.4 MAC layer, through the implementation of the corresponding primitives. Furthermore, we implemented a processor emulator for an actual 802.15.4 device, namely the Jennic JN5139. The processor emulator allows running in real time on the emulation testbed the same binary firmware that is executed on the real JN5139 devices. The sensing functionality of JN5139 is also emulated for temperature, humidity, and light level sensors.

Our emulation testbed enables realistic experiments with IEEE 802.15.4 networked systems. Its operation was validated through several series of experiments, both in static and mobile scenarios. The differences between the results obtained in real-world trials and those of emulation experiments did not exceed more than a few percents.

A current limitation of our testbed is that it doesn't consider the power consumption of the sensor nodes. Therefore, we intend to add such support, which is important in scenarios with battery-operated sensors. In addition, we envisage exploring in more detail the challenges related to performing experiments that combine real and emulated devices, which we consider very promising.

Acknowledgments

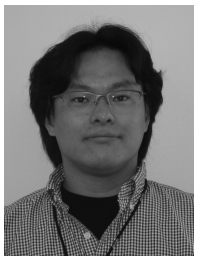
The authors are grateful to Takashi Okada for his support related to the use of the JN5139 devices. We also thank Yuki Hirakawa and Prof. Ikuko Yairi for their assistance in performing some of the real-world trials presented here, and for the 3D modeling required in our experiments.

References

[1] M. Beigl, SensorRAUM, http://www.duslab.de/sensorraum/.
[2] R. Beuran, J. Nakata, T. Okada, T. Kawakami, K. Chinen, Y. Tan, and Y. Shinoda, Emulation framework for the design and development of active RFID tag systems, Journal of Ambient Intelligence and Smart Environments (JAISE), IOS Press, vol. 2, no. 2, April 2010, pp. 155-177.
[3] R. Beuran, J. Nakata, T. Okada, L. T. Nguyen, Y. Tan, and Y. Shinoda, A Multi-purpose Wireless Network Emulator: QOMET, IEEE Intl. Conf. on Advanced Information Networking and Applications (AINA 2008), FINA 2008 symposium, Okinawa, Japan, March 25-28, 2008, pp. 223-228.
[4] R. Beuran, J. Nakata, Y. Tan, and Y. Shinoda, IEEE 802.15.4 Network Emulation Testbed, IEEE Intl. Conf. on Advanced Information Networking and Applications (AINA 2011), Biopolis, Singapore, March 22-25, 2011, pp. 451-458.
[5] R. Beuran, L. T. Nguyen, T. Miyachi, J. Nakata, K. Chinen, Y. Tan, and Y. Shinoda, QOMB: A Wireless Network Emulation Testbed, IEEE Global Communications Conference (GLOBECOM 2009), Honolulu, Hawaii, USA, November 30-December 4, 2009.
[6] Crossbow Technologies, MICA2 Wireless Modules, http://www.xbow.com.
[7] D. B. Faria, Modeling signal attenuation in IEEE 802.11 wireless LANs, Technical Report TRKP06-01187, Stanford University, Palo Alto, California, USA, July 2005.
[8] IEEE Standard 802.15.4-2006: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (LR-WPANs), 2006.
[9] Jennic Ltd., JN5139 Wireless Microcontroller, http://www.jennic.com/

products/wireless_microcontrollers/jn5139.
[10] D. Johnson, T. Stack, R. Fish, D. M. Flickinger, L. Stoller, R. Ricci, and J. Lepreau, Mobile Emulab: A robotic wireless and sensor network testbed, IEEE INFOCOM 2006, Barcelona, Spain, April 23-29 2006.
[11] P. Levis, N. Lee, M. Welsh, and D. Culler, TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications, ACM Conf. on Embedded Networked Sensor Systems (SenSys'03), Los Angeles, California, USA, November 5-7, 2003, pp. 126-137.
[12] R. W. Lewis, P. Nithiarasu, and K. N. Seetharamu, Fundamentals of the finite element method for heat and fluid flow, Wiley, 2004.
[13] T. Miyachi, K. Chinen, and Y. Shinoda, StarBED and SpringOS: Large-scale General Purpose Network Testbed and Supporting Software, Intl. Conf. on Perf. Evaluation Methodologies and Tools (Valuetools 2006), ACM Press, Pisa, Italy, October 2006.
[14] J. Nakata, T. Miyachi, R. Beuran, K. Chinen, S. Uda, K. Masui, Y. Tan, and Y. Shinoda, StarBED2: Large-scale, Realistic and Real-time Testbed for Ubiquitous Networks, TridentCom 2007, Orlando, Florida, USA, May 21-23, 2007.
[15] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, Cross-level sensor network simulation with COOJA, IEEE Intl. Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2006), Tampa, Florida, USA, November 2006.
[16] J. Polley, D. Blazakis, J. McGee, D. Rusk, and J. S. Baras, ATEMU: A Fine-grained Sensor Network Simulator, IEEE Conf. on Sensor and Ad Hoc Communications and Networks (SECON 2004), Santa Clara, California, USA, October 4-7, 2004, pp. 145-152.
[17] T. S. Rappaport, Wireless Communications: Principles and Practice, Prentice Hall PTR, 2nd edition, 2002.
[18] Y. Tan, Home Network Technology 2011 for Smart House (in Japanese), Impress R&D, 2011.
[19] Texas Instruments Inc., CC2531: System-on-Chip Solution for IEEE 802.15.4 and ZigBee Applications, http://focus.ti.com/docs/prod/folders/print/cc2531.html.
[20] University of Utah, School of Computing, Emulab - Total network testbed, http://www.emulab.net.
[21] G. Werner-Allen, P. Swieskowski, and M. Welsh, MoteLab: a wireless sensor network testbed, Intl. Symp. on Information Processing in Sensor Networks (IPSN 2005), Los Angeles, California, USA, April 25-27, 2005, pp. 483- 488.
[22] J. Zhou, Z. Ji, and R. Bagrodia, Twine: A hybrid emulation testbed for wireless networks and applications, IEEE INFOCOM 2006, Barcelona, Spain, April 23-29 2006.

Razvan Beuran received the B.Sc., M.Sc. and Ph.D. degrees from "Politehnica" University, Bucharest, Romania in 1999, 2000 and 2004, respectively (the PhD degree was delivered jointly with "Jean Monnet" University, Saint Etienne, France). From 2001 to 2005 he was with CERN, Geneva, Switzerland as a researcher. Since 2006 he is researcher with the National Institute of Information and Communications Technology, Hokuriku StarBED Technology Center, Ishikawa, Japan. Since 2007 he is also project researcher with the Japan Advanced Institute of Science and Technology, Ishikawa, Japan. His research topics include network dependability studies in wired and wireless networks, in particular through the use of network emulation. He is a member of IEEE.



Junya Nakata received his Ph.D. in Information Science from Japan Advanced Institute of Science and Technology in 2009. Since 2012 he is assistant division director for the engineering headquarters of Uniden Corporation. Before joining Uniden Corporation, he worked for several organizations, such as Keio University as project associate professor, National Institute of Information and Communications Technology as researcher, and Nokia Corporation as research engineer.

Yasuo Tan was born in 1965. He received his Ph.D. from Tokyo Institute of Technology in 1993. He joined Japan Advanced Institute of Science and Technology (JAIST) as an assistant professor of the School of Information Science in 1993. He has been a professor since 1997. He is interested in Ubiquitous Computing Systems especially Home Networking Systems. He is a leader of Residential ICT SWG of NeW Generation Network Forum, a chairman of Green Grid Platform at Home Alliance, an advisory fellow of ECHONET Consortium, and a member of IEEE, ACM, IPSJ, IEICE, IEEJ, JSSST, and JNNS.

Yoichi Shinoda received his B.E., M.E. and Ph.D. from Tokyo Institute of Technology in 1983, 1985 and 1989, respectively. He joined Japan Advanced Institute of Science and Technology in 1991 as a professor of the School of Information Science. His research interests include distributed and parallel computing, networking systems, operating systems, and information environments.