2011 International Conference on Advanced Information Networking and Applications

IEEE 802.15.4 Network Emulation Testbed

Razvan Beuran*, Junya Nakata*, Yasuo Tan† and Yoichi Shinoda†
*National Institute of Information and Communications Technology
Nomi, Ishikawa, Japan
Email: razvan@nict.go.jp, jnakata@starbed.org
†Japan Advanced Institute of Science and Technology
Nomi, Ishikawa, Japan
Email: {ytan, shinoda}@jaist.ac.jp

***Abstract*—IEEE 802.15.4 networks are promising solutions for wireless personal area networks, and in particular for wireless home area networks. IEEE 802.15.4 has numerous applications in fields such as energy management and home automation. However, real-world trials with 802.15.4 devices are difficult because of the characteristics of these devices (small dimensions, wireless communication), and the potentially large size of the network. We present in this paper an IEEE 802.15.4 network emulation testbed that makes possible repeatable and controllable live experiments with 802.15.4-based devices. The testbed is built by extending the functionality of the wireless network emulation testbed named QOMB with 802.15.4 PHY and MAC layer capabilities, as well as 802.15.4 device processor emulation. We illustrate the usability of the 802.15.4 network emulation testbed with a case study of home networking used for automation related to environment control.**

***Keywords*-IEEE 802.15.4; network emulation; processor emulation; network testbed.**

I. Introduction

IEEE 802.15.4 is a standard for the physical and media access control layers of low-rate wireless personal area networks (LR-WPANs). The standard targets the low-cost, low-speed ubiquitous communication between devices. IEEE 802.15.4 is also the basis of higher-layer specifications such as ZigBee that add extra encryption, authentication, and application services. Thus, 802.15.4 and ZigBee are becoming the solution of choice for various home network applications, such as energy management, remote control, home automation, and health care. The popularity of IEEE 802.15.4 lead to significant R&D efforts related to this standard in the fields of embedded systems and networking, ubiquitous intelligence, and pervasive computing.

The reduced size of 802.15.4-based devices, the wireless nature of their communication, and the large number of devices typically required by a home network application all make that real-world trials with 802.15.4 networks are difficult to perform. The often-used alternative of network simulation, while meeting some of the requirements, raises questions regarding the realism of the experiments, and does not typically allow to directly execute the device firmware.

In this paper we present an IEEE 802.15.4 network emulation testbed that allows conducting realistic live experiments with 802.15.4 networks in a repeatable manner by running the real device firmware in controlled and reproducible network conditions. The emulation testbed is based on the framework of a generic wireless network emulation testbed that we designed and developed, named QOMB. In this work QOMB was extended for the purpose of 802.15.4 network emulation through the implementation of specific functionality, such as IEEE 802.15.4 PHY and MAC layer emulation, as well as 802.15.4 device processor emulation.

We illustrate the usage of our 802.15.4 network emulation testbed with a case study of home networks for sensing applications that could be used in connection with automation for temperature regulation, and potentially also for smart energy management. This case study demonstrates the usability of our testbed, and its capabilities for the performance evaluation and validation of 802.15.4 networks in complex environments, including mobility.

The contributions of this paper are as follows:

- A probabilistic model for the 802.15.4 PHY layer;
- The implementation of an 802.15.4 device emulator, including the MAC layer and the device processor;
- An 802.15.4 network case study of a temperature sensing application.

The paper is organized as follows. In Section II we summarize the main aspects related to QOMB, the framework on which our testbed is built. In Section III we detail the additional features that we implemented in order to support 802.15.4 network emulation on QOMB. We then illustrate the use of the testbed for home automation in Section IV. The paper ends with a section on related work (Section V), and conclusions (Section VI).

II. QOMB Wireless Network Emulation Testbed

QOMB is a wireless network emulation testbed that was initially created for IEEE 802.11 (WLAN) network emulation. The general architecture of QOMB, and utilization examples in the context of WLAN were presented in [4]. The modular architecture of QOMB and its components makes it possible to extend the testbed to other wireless network technologies and devices in a straightforward manner. By adding the necessary features, we could employ QOMB for

1550-445X/11 $26.00 © 2011 IEEE
DOI 10.1109/AINA.2011.25

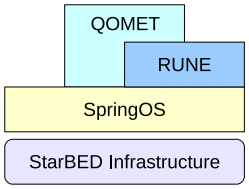

Figure 1. Logical hierarchy of QOMB components.

the performance analysis of a pedestrian localization system using active RFID tags, as it was detailed in [2].

In this section we shall introduce the main components of QOMB, and the generic architecture used on QOMB for ubiquitous system emulation. In particular, the QOMB wireless network emulation testbed is realized by integrating the wired-network testbed that is StarBED (and its support tools, SpringOS and RUNE) with the wireless network emulation set of tools that is QOMET. The logical hierarchy of the elements that compose QOMB is shown in Figure 1.

A. StarBED

StarBED is a large-scale wired-network testbed at the Hokuriku Research Center of the National Institute of Information and Communications Technology, located in Ishikawa, Japan [8]. With more than 1000 interconnected PCs available for experiments, StarBED makes possible a wide range of network experiments, and represents the infrastructure of QOMB.

1) SpringOS: The main experiment-support software tool for StarBED is called SpringOS, and allows users to easily perform complex experiments with a large number of hosts [8]. The main two functions of SpringOS are:

- Experiment preparation: configure the experiment hosts and network so that they are ready for experiment execution;
- Experiment execution: effectively carry out the experiment by executing in the required order the necessary commands on the experiment hosts.

SpringOS is oriented towards IP network experiments, therefore in our work related to ubiquitous systems and in particular to 802.15.4 networks, which do not employ IP communication, we used another support tool: RUNE.

2) RUNE: The experiment-support software tool that is aimed at making possible ubiquitous network emulation experiments on StarBED is called RUNE (Real-time Ubiquitous Network Emulation environment). RUNE is a framework for performing emulation experiments with a large number of emulated devices [9]. Different from SpringOS, RUNE has no restrictions regarding the type of network used in experiments. Hence, RUNE is the ideal choice for driving 802.15.4 system emulation on StarBED, as these devices do not use IP communication.

For the purpose of this paper we summarize only the most important aspects related to RUNE. First of all, RUNE experiments are globally managed by a module called *RUNE Master* executed on a master computer that can be any StarBED host. RUNE Master communicates with modules called *RUNE Manager*, an instance of which is executed on each experiment host. The functionality of the emulated ubiquitous devices is reproduced by entities that are generically called *spaces* in RUNE. Spaces on an experiment host are controlled by the local RUNE Manager. They communicate with each other via abstractions of communication channels called *conduits*. Note that conduits are only logical channels for passing messages between spaces with the assistance of RUNE Managers, and network emulation has to be performed by using dedicated spaces for communication condition emulation, as shown in Section II-C.

B. QOMET

QOMET (Quality Observation and Mobility Experiment Tools) is a set of tools for wireless network emulation in complex scenarios including mobility. QOMET provides the necessary mechanisms for performing wireless network emulation in a distributed manner by reproducing the communication conditions between the current node and the other nodes in the experiment. Note that QOMET is not an emulator *per se*, and cannot be executed in a standalone manner. Instead, QOMET relies on the experiment management mechanisms of StarBED for its distributed execution. This is why the integration of all the tools that we mentioned so far effectively creates a new entity, the wireless network emulation testbed QOMB.

The most important components of QOMET are the libraries called *deltaQ* and *chanel*. Another library, called *wireconf*, is only used in the context of IP network emulation, hence it will not be discussed in this paper.

1) DeltaQ library: The deltaQ library is in charge of computing the communication conditions between wireless nodes given a user-defined scenario. The scenario specifies the properties of the wireless nodes (position, network technology parameters, mobility patterns, etc.), 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 scenario, in which the wireless nodes move and communicate with each other.

2) Chanel library: The communication conditions computed by deltaQ are recreated during the live experiment by the chanel library. This library is in charge of delivering the messages from a wireless node to all the other nodes with which it can communicate according to the scenario, after applying the corresponding network degradation (packet loss, delay, bandwidth limitation).

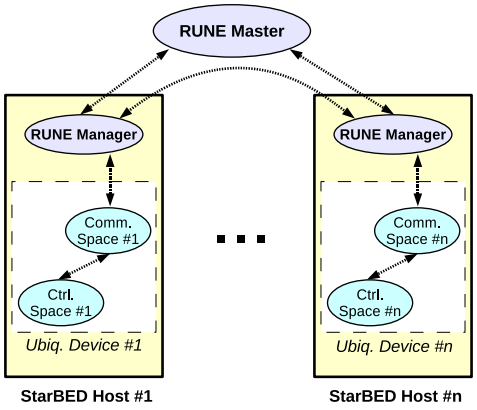

Figure 2. Architecture of the QOMB ubiquitous network emulation testbed framework.

C. Emulation Architecture

In Figure 2 we show the architecture of the QOMB ubiquitous network emulation framework that is created by combining the previously mentioned components. Note how RUNE manages the experiment executed on StarBED hosts via the global RUNE Master and the local RUNE Manager modules (SpringOS is not mentioned in the figure because it only plays a role in experiment preparation, as indicated in Section II-A). Each logical ubiquitous device is in practice composed of two elements, which are implemented as RUNE spaces: a communication space and a control space.

The communication space recreates the communication conditions between each emulated ubiquitous device and the other devices in the experiment. Practically, the communication space employs the chanel library to accomplish this task. In its turn, the chanel library uses the communication conditions computed by the deltaQ library. For 802.15.4 experiments, the QOMET deltaQ library had to be enhanced by adding support for IEEE 802.15.4 PHY layer emulation. These modifications will be discussed in Section III-A.

The control space reproduces the behavior of the emulated ubiquitous device. Its most important part is represented by the processor emulator, that emulates the execution of a real ubiquitous device processor, so as to allow using on QOMB the same firmware with the real ubiquitous device. As processors are specific to the hardware whose behavior is reproduced on QOMB, such a processor emulator needs to be implemented for each type of ubiquitous device. Other modules and functionality may be necessary too, depending on the emulated device and type of experiment. All these issues will be analyzed in Section III-B.

III. IEEE 802.15.4 Network Emulation

In order to add IEEE 802.15.4 network emulation capabilities to QOMB, the two generic spaces used for ubiquitous device emulation that were presented in Figure 2, the communication space and the control space, need to support 802.15.4 specific functionality. While some of this functionality is generic, such as the 802.15.4 PHY layer emulation capability integrated with the QOMET deltaQ library, some of this functionality is necessarily device dependent. This is because the goal of executing real device firmware in the emulator requires considering particular devices.

In this work we focused on an 802.15.4 device named JN5139, that is manufactured by Jennic, Ltd. 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 192kB ROM for system code, including protocol stack, and 96kB RAM for system data and optional program code. The device has other features such as comparators, timers, counters, and various kinds of interfaces. In particular, the device that we emulate has on-board temperature, light level and humidity sensors, and an LCD display.

A. Communication Space

The generic communication space that was presented in Figure 2 was extended so as to reproduce 802.15.4 communication conditions between the wireless nodes. For realism purposes, we decided to execute an 802.15.4 MAC layer implementation in the complementary control space; hence, only the 802.15.4 PHY layer had to be modeled in the communication space. This model was created by adapting the 802.11 PHY layer model already existing in QOMET [3] to meet the specifications of the 802.15.4 standard. This model was integrated with the deltaQ library of QOMET.

The goal of the model is to compute the frame error rate, delay and bandwidth limitation that should be applied to the frames that are received from the control space before sending them to the other emulated devices in order to recreate 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.

Propagation effects are considered through the use of the log-distance path loss model [11], which gives the received power at a distance d, P_r, as function of the received power at the distance of 1 m, P_{r0}, the attenuation coefficient α, the wall attenuation W, and the standard deviation σ of the shadowing component, as shown in Equation (1):

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

Note that P_{r0} depends on the transmit power P_t as given by Equation (2) below, where c is the speed of light, and F is the frequency of the electromagnetic waves used:

$$P_{r0} = P_t - 20 \cdot log_{10}\frac{4\pi \cdot F}{c}. \quad (2)$$

The frame error rate is then probabilistically computed from the received power by using Equation (3):

$$FER_S = FER_{S0} \cdot e^{S-(Pr-N)-N_{th}}, \quad (3)$$

where S is the receive sensitivity threshold of the device transceiver (provided by 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 standardized frame size, that we denote by FS_S. Therefore, Equation (3) gives a result that is specific to this frame size (which is equal to 20 bytes for 802.15.4). To extend the calculation of the frame error rate to frames with arbitrary sizes, the following equation is necessary:

$$FER = 1 - (1 - FER_S)^{FS/FS_S}, \quad (4)$$

where FS denotes the size of the frame for which the error rate 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 in our model. In particular, the equation we use to compute the transmission delay, D, is:

$$D = T_{PHY} + T_{PSDU} + T_{IFS}, \quad (5)$$

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. As for T_{PSDU}, it is computed by:

$$T_{PSDU} = (8 \cdot F_{PSDU})/R, \quad (6)$$

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

Finally, the amount of effective bandwidth that is available at the PHY layer is:

$$B = \frac{T_{PHY} + T_{PSDU}}{D} \cdot R. \quad (7)$$

The computed frame error rate, delay, and bandwidth (FER, D, and B, respectively) are used by the communication space to recreate conditions in the 802.15.4 emulated network that correspond to the user-defined scenario. Table I shows the model parameter values that are used for the above equations when performing the experiments that will be described in Section IV.

B. Control Space

The generic control space that was presented in Figure 2 needs specific functionality related to JN5139 devices in order to be used for their emulation. The modules required in order to provide this functionality will be described below.

Table I
PARAMETER VALUES FOR IEEE 802.15.4 PHY EMULATION

Parameter	Value
Transmit power, P_t	1 dBm
Attenuation coefficient, α	4.02
Shadowing parameter, σ	0.0
Wall attenuation, W	9.6 dBm
Frequency, F	2.4425 GHz
Speed of light, c	$2.9979 \cdot 10^8$ m/s
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 kbps

1) Processor Emulator: One of the most important requirements for enabling realistic emulation experiments is in our view the ability to execute the same firmware with the real devices. Therefore, an emulator was implemented for the processor of the JN5139 device, so that its firmware can be executed directly on our emulation testbed.

The processor emulator that we implemented, called ORE (OpenRISC Emulator), supports the hardware components of the JN5139 micro-controller, including memory and counters, but also the other features, such as sensors. The only point that is not supported directly is the 802.15.4 transceiver, and the 802.15.4 MAC functionality that is associated to it. Fully implementing the transceiver would have made little sense, since we emulate the PHY layer in the communication space, as mentioned in Section III-A.

The trade-off we faced was the following: (i) implement the transceiver hardware emulation *only* so as to run the Jennic proprietary 802.15.4 MAC implementation; or, (ii) avoid implementing the transceiver, and instead implement an equivalent for the 802.15.4 MAC functionality. The main drawback of the first solution is the performance penalty that would occur when executing the proprietary MAC implementation on the emulated device processor.

As a result, we decided to proceed with the second solution: an alternative 802.15.4 MAC implementation that runs as a separate module in the control space, and is executed *natively* on the CPU of the experiment host. Thus, the processor emulator will handle all firmware instructions except those related to MAC operation, which are passed to this separate module. The resulting hierarchy of the two components of the control space is shown in Figure 3, which also represents the binary firmware running on top of the processor emulator.

2) IEEE 802.15.4 MAC Emulator: As a result of the decision explained above, whenever an application tries to call an 802.15.4 MAC primitive, the ORE processor emulator intercepts it, and passes to the 802.15.4 MAC emulator module that each ORE instance manages (cf. Figure 3). The MAC emulator module is executed as native binary code

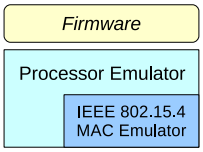

Figure 3. Hierarchy of the control space components.

on the experiment host on which emulation takes place, hence with a speed that is much faster than the execution speed of ORE (for instance, about 100 times faster when running on an 1.6 GHz computer CPU versus the 16 MHz RISC processor). As a result, the performance of the library is significantly improved, and our emulation testbed can support seamlessly the live execution of 802.15.4 application firmware. We note that because of the faster execution of the library, the MAC will actually run faster on our testbed that it would do on JN5139 devices. This difference is nevertheless compensated by the absolute delays built into the MAC protocol, which makes that overall the emulator produces and consumes messages in real time and in synchronization with the wall clock, albeit only a loosely-coupled synchronization compared to that of the real devices.

3) Temperature sensor model: The JN5139 device has on-board temperature, light level, and humidity sensors, which are all supported by the emulator from the point of view of interaction with them at hardware level. Nevertheless, in order to perform realistic experiments it was necessary to also model the behavior of the sensors themselves.

At this point we only integrated with the control space a model for the temperature sensor, but models for the other sensors will be added in the near future. In particular, the equation we use for computing the temperature reading of the sensor at time t after the sensor is brought into a certain environment, that we denote by $T(t)$ is:

$$T(t) = T_A - (T_A - T_0)e^{-t/C_{th}}, \qquad (8)$$

where T_A is the environment temperature to which the sensor is adapting, T_0 is the initial temperature of the sensor, and C_{th} is the thermal time constant of the sensor, for which a typical value according to our survey is 10 s.

C. Discussion

We acknowledge the fact that some of the sensor characteristics that we use as parameters of our models may not be 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.

One other issue is time synchronization. Note that we use no specific distributed time synchronization mechanism in our testbed. Instead, where needed, each 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 ensures sufficient time accuracy for our purposes.

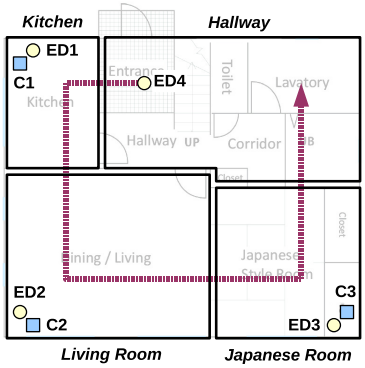

Figure 4. Emulation experiment scenario.

IV. Experimental Results

To illustrate the capabilities of our 802.15.4 network emulation testbed we consider a home network scenario. The scenario is based on the characteristics of a real house that has been built in the vicinity of our research center for the purpose of conducting real-world trials with home networks. The relation between the real house and our 802.15.4 network emulation testbed is bidirectional. On one hand, the real house is intended as a validation environment for our emulation testbed. On the other hand, the testbed is aimed at extending the scale of the experiments that can be performed in the real house. For instance, one could use more sensors in the virtual environment than physically available in the real house.

A. Experiment Setup

The proof of concept experiments that we present involve two types of 802.15.4 devices, namely:

- Coordinators (C) that form the root of the 802.15.4 network star topology, and could bridge to other networks;
- End devices (ED) that execute the 802.15.4 application, and connect to the root node.

The emulation experiment we designed is set up in a virtual environment based on the ground floor of the real house, which includes three rooms, “Kitchen”, “Living Room”, and “Japanese Room”, as shown in Figure 4. The ground floor also includes several smaller spaces, such as a hallway, corridor, and lavatory, that we designated in the figure as a forth area with the generic name “Hallway”.

In our experiment we considered that each of the three rooms has an independent network, but they share the network identifier. Each room includes a coordinator, and also a static end device, denoted in Figure 4 by C1 to C3, and ED1 to ED3, respectively. Transceiver and environment properties

were chosen so that there is no interference between these three networks.

The functionality of the JN5139 application that we used on the end devices is to measure the temperature, light level, and humidity, and report them regularly to the coordinator to which they are associated. In the experiment we assume that the coordinators are all connected by a wired network to a central computer that will manage the entire house for purposes of ambient control, such as temperature and lighting adjustments, by controlling systems such as air conditioners and lights.

Note that more advanced functions could be provided as well, such as energy management. Thus, the central computer could select in an intelligent manner the most energy-efficient way to achieve a certain goal. For temperature regulation, for instance, several choices can be made: whether to turn on or off the air conditioner, to open or close the windows, to pull up or down the blinds, depending on the internal and external temperatures, time of day, presence of people in the room, etc. Although this functionality is still only hypothetical, we want to stress the fact that the remote control of the above mentioned devices is possible in the real house that we use for trials, and such applications are envisaged for the near future.

In addition to the six static 802.15.4 devices discussed so far, we included in the emulation experiment a seventh one, an end device which is mobile. Figure 4 shows the initial position of the mobile end device, denoted by ED4, and also its trajectory starting from the hallway area (which has no network). As the three networks in the house share the same network identifier, the mobile device can connect to any of them which is in its communication range.

The mobile end device executes the same application as the static end devices. The role of the mobile device is to illustrate the mobility emulation capabilities of the testbed, and is intended to represent any kind of mobile sensor; it could also be indirectly used for presence detection, as its messages or lack thereof would notify the system of the fact whether the sensor is present or not in a room.

One target for these experiments is to validate the overall functioning of the 802.15.4 emulator. As the application used is performing sensing, and the temperature sensors of JN5139 are fully supported in our emulator, we focused on the temperature sensing functionality. The conditions of the experiment from this point of view are given in Table II, which shows the fixed temperatures that we assigned to each room (expressed in Celsius degrees), as well as the time interval during which the end device ED4 is present in each of the areas (as the node starts and finishes moving in the hallway, two intervals are shown for this area).

B. Sensing Functionality

To evaluate the sensing functionality as reproduced on the 802.15.4 emulation testbed we collected the temperature data

Table II
EXPERIMENT CONDITIONS FOR TEMPERATURE SENSING

Area name	Temperature [°C]	ED4 presence interval [s]
Hallway	24	0–122 and 306–360
Kitchen	28	122–184
Living Room	26	184–249
Japanese Room	27	249–306

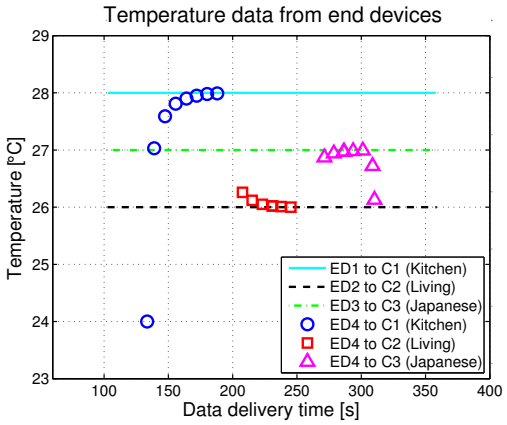

Figure 5. Temperature data values as received by the three coordinators.

received during the experiment by the three coordinators, C1, C2, and C3, from the four end devices, ED1, ED2, ED3, and ED4. The data is plotted in Figure 5. Note that, although the total experiment duration was 360 s, we do not take into account the first 100 s of the experiment, which include the initialization of the emulated devices. Moreover, motion of ED4 starts at time 120 s.

Figure 5 shows that each static sensor reports the correct temperature value for the virtual area in which it is located, as follows: ED1 sensed the temperature of the kitchen (28 °C), ED2 sensed the temperature of the living room (26 °C), and ED3 sensed the temperature of the Japanese room (27 °C). As we assumed the temperatures for each area to be fixed, sensor data from static sources is also constant during the entire experiment duration.

The mobile end device, ED4, goes through all the three virtual rooms during its motion. As a consequence, sensor data will be sent to the coordinator to which ED4 is associated in the room in which it is located during a certain period. Hence, the data will be sent to C1, placed in the kitchen, in the first part of the trajectory, then to C2, placed in the living room, in the second part, and finally to C3, placed in the Japanese room, at the end. Note how sensor readings for ED4 change in Figure 5 starting at the time the emulated mobile device enters a certain room, as it adapts to the ambient temperature of a room. Thus, the sensor data provided by ED4 reaches 28 °C while the device is in the kitchen (around time 180 s), then goes to 26 °C after the device arrives in the living room (around time 230 s), and raises to 27 °C while the mobile device is in the Japanese

room (around time 290 s). The last two sensor data values (after time 300 s), received as ED4 entered the hallway, and before the communication with C3 was interrupted, show how the temperature sensor begins adapting to the temperature of the hallway, which is 24 °C.

Figure 5 demonstrates that, even when being executed on our 802.15.4 emulation testbed, the temperature sensing application functions in the same manner with the case when it would be executed in reality. This means that the testbed can be employed as a replacement of real-world trials, for instance in the development of an intelligent environment control system that uses the data provided by the sensors to make the required decisions. Moreover, the testbed could be employed for assessing 802.15.4 applications that are in the development process, to evaluate whether they behave as expected, and what are their performance characteristics.

C. Hand-off Procedure

We use the same experiment described above to show how our testbed can be used to measure characteristics of the emulated ubiquitous system that may be difficult to determine by real-world trials. Such a parameter is the hand-off delay. As the end device ED4 moves from room to room, it will have to connect to the network in each room in order to successfully send its sensor data. Measuring the hand-off delay in the equivalent real scenario is difficult because it requires capturing the key packets in each room and recording their time, while making sure that all the capture devices are time synchronized with each other.

However, all our emulated devices are time synchronized by design, as the hosts they are executed on are synchronized using the NTP protocol, and all device initialization is essentially simultaneous given the architecture of RUNE. Therefore, such time measurements can be made in a straightforward manner on the emulation testbed from the log files of each subsystem.

Table III presents the hand-off delay results for ED4 that we calculated for a series of five experiments performed in the same conditions as above. We show the minimum, average, maximum, and standard deviation results separately for each hand-off event, i.e, first when ED4 transitions between C1 and C2, and then between C2 and C3, but also as a global average with both hand-offs included. We note that the results are reasonably stable for the first hand-off, around 19 s. For the second hand-off, while results are typically around 18 s, in one case hand-off succeeded with one packet faster than usual, resulting in a best-case hand-off of about 10 s; this gave a smaller average of around 16 s, but also increased the standard deviation for that hand-off.

Table III
HAND-OFF DELAY RESULTS

	Hand-off delay			
Conditions	Min. [s]	Avg. [s]	Max. [s]	Std.
Hand-off from C1 to C2	19.35	19.51	19.66	0.15
Hand-off from C2 to C3	10.06	16.24	18.60	3.53
Both hand-offs included	10.06	17.88	19.66	2.92

V. RELATED WORK

To the best of our knowledge, there exists currently no system which has an identical functionality with our IEEE 802.15.4 network emulation testbed. There are, however, several approaches and tools that are related to our research.

SensorRAUM is a project whose goal is to transfer the physical environment into a quasi-realistic virtual representation within a computer [1]. This representation makes possible the interaction between real sensors and the virtual world. We also emulate the sensors themselves, giving the user more control over the experiment.

TOSSIM is a TinyOS mote simulator targeting the development of sensor network applications [7]. Its authors claim that it scales to thousands of nodes, and compiles directly from TinyOS code. Hence, developers can test not only algorithms, but also their implementations, albeit through simulation. ATEMU is a software emulator for systems based on the Atmel AVR processor [10]. It also includes support for other peripheral devices on the MICA2 sensor node platform [5], such as the radio. ATEMU can be used to conduct studies in a controlled simulation environment, and is compatible at binary level with the MICA2 hardware. Both TOSSIM and ATEMU are essentially simulators focusing on particular processors, and use only basic modeling for the wireless communication (for instance, ATEMU only supports free-space propagation). Moreover, they provide no guarantee as to how far from wall clock time the experiment execution is, since everything is performed in logical time, which is typically much slower than real time for large-scale experiments. Our testbed has none of these drawbacks.

Emulab provides a sensor network testbed including 25 MICA2 motes [12]. All motes are equipped with a serial port, for control and debugging purposes, and can be used remotely for experiments. Another sensor testbed is MoteLab [13], 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 interferences. Moreover, no mobility experiments are possible on these testbeds. Mobile Emulab uses robots to achieve reproducibility of the motion of the wireless nodes [6]. However, the communication is still subject to potential undesired influences. In addition, the range and speed of the mobile nodes are limited.

Closer to our approach are the wireless network emulation testbeds, mainly related to IEEE 802.11 networks. A system such as TWINE uses computer models to perform real-time experiments [14]. Thus it avoids undesired interferences and

side effects, in a similar manner to the approach used by QOMB for IEEE 802.15.4. However, QOMB implemented not only 802.15.4 network emulation, but also the emulation of the hardware of real 802.15.4 devices.

VI. Conclusion

In this paper we presented an IEEE 802.15.4 network emulation testbed that can be used to perform experiments with 802.15.4 applications in a controlled virtual environment. The testbed is built by extending the functionality of an existing wireless network emulation testbed called QOMB, that was created by the integration of a 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 existing 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. Combined with the 802.15.4 network support, this effectively transforms QOMB into a testbed that makes possible realistic 802.15.4 network emulation experiments.

To illustrate the functionality of the 802.15.4 network emulation testbed we presented an experiment using a scenario based on the topology of a real house that is available for ubiquitous network experiments in the vicinity of our research center. The experiment demonstrated the successful emulation of features of 802.15.4-based systems such as temperature sensing, which can be used for home automation and environment control. The emulated mobility capabilities also made possible to measure characteristics of the 802.15.4 network such as the hand-off delay. Thus, we emphasized how the testbed can be used to assess the performance of 802.15.4 applications.

Our future work will focus on several research directions. Ongoing work refers to validating our testbed by using 802.15.4 nodes in the real house for trials, and comparing those results with equivalent experiments executed on the emulation testbed, including for large-scale scenarios. A current limitation of our testbed is that it doesn't consider some sensor characteristics, such as power consumption; therefore, we intend to add support for these type of characteristics as well, which are important in scenarios with battery-operated sensors. Another direction refers to implementing the necessary components for making complete home automation experiments, such as the communication with the actuators, and an algorithm for environment control.

References

[1] M. Beigl, *SensorRAUM*, http://www.duslab.de/sensorraum/.

[2] R. Beuran, J. Nakata, T. Okada, T. Kawakami, K. Chinen, Y. Tan, 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, Y. Shinoda, *A Multi-purpose Wireless Network Emulator: QOMET*, 22nd 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, L. T. Nguyen, T. Miyachi, J. Nakata, K. Chinen, Y. Tan, Y. Shinoda, *QOMB: A Wireless Network Emulation Testbed*, IEEE Global Communications Conference (GLOBECOM 2009), Honolulu, Hawaii, USA, November 30-December 4, 2009.

[5] Crossbow Technologies, *MICA2 Wireless Modules*, http://www.xbow.com.

[6] D. Johnson, T. Stack, R. Fish, D. M. Flickinger, L. Stoller, R. Ricci, J. Lepreau, *Mobile Emulab: A robotic wireless and sensor network testbed*, IEEE INFOCOM 2006, Barcelona, Spain, April 23-29 2006.

[7] P. Levis, N. Lee, M. Welsh, D. Culler, *TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications*, ACM Conf. on Embedded Networked Sensor Systems (SenSys'03), Los Angeles, California, U.S.A., November 5-7, 2003, pp. 126-137.

[8] T. Miyachi, K. Chinen, Y. Shinoda, *StarBED and SpringOS: Large-scale General Purpose Network Testbed and Supporting Software*, Intl. Conf. on Performance Evaluation Methodologies and Tools (Valuetools 2006), ACM Press, Pisa, Italy, October 2006.

[9] J. Nakata, T. Miyachi, R. Beuran, K. Chinen, S. Uda, K. Masui, Y. Tan, Y. Shinoda, *StarBED2: Large-scale, Realistic and Real-time Testbed for Ubiquitous Networks*, 3rd Intl. Conf. on Testbeds and Research Infrastructures for the Development of Networks & Communities (TridentCom 2007), Orlando, Florida, U.S.A., May 21-23, 2007.

[10] J. Polley, D. Blazakis, J. McGee, D. Rusk, 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, U.S.A., October 4-7, 2004, pp. 145-152.

[11] T. S. Rappaport, *Wireless Communications: Principles and Practice*, Prentice Hall PTR, 2nd edition, 2002.

[12] University of Utah, School of Computing, *Emulab - Total network testbed*, http://www.emulab.net.

[13] G. Werner-Allen, P. Swieskowski, M. Welsh, *MoteLab: a wireless sensor network testbed*, Intl. Symp. on Information Processing in Sensor Networks (IPSN 2005), Los Angeles, California, U.S.A, April 25-27, 2005, pp. 483- 488.

[14] J. Zhou, Z. Ji, R. Bagrodia, *Twine: A hybrid emulation testbed for wireless networks and applications*, IEEE INFOCOM 2006, Barcelona, Spain, April 23-29 2006.