Model-Based QoS Evaluation and Validation for Embedded Wireless Sensor Networks

Sven Jäger, Tino Jungebloud, Ralph Maschotta, and Armin Zimmermann

***Abstract*—Wireless communication technologies have the potential to significantly reduce cabling cost, cut down on weight, and simplify the overall structure of industrial embedded systems. Potential risks and costs of a technological switch from wired interconnections toward a wireless solution are difficult to estimate, however. Model-based system engineering can help to reduce risk and cost and to increase the quality of the resulting system. However, standard network simulation software has to be adapted to the particular requirements of embedded wireless sensor networks such as system operational modes and phases, comprehensive means of configuration, and performance analysis; and models are not easy to validate in detail. This paper presents a highly scalable model for industrial embedded wireless networks to validate different system configurations and applies it to a system setup designed for avionic environments, using a software architectural framework for domain-specific simulation-based tools. The presented model is detailed enough to cover the significant behavior for quality of service requirements, including delays, throughput, reliability, energy consumption, and node lifetime. An additional important contribution is the direct connection between design tool and test bed or prototype system, allowing a seamless configuration of a hardware setup and an integration of measured behavior for a fine-grained validation of model and prototype. The application to a real-life system setup shows the benefits of the integrated approach.**

***Index Terms*—Energy consumption, modeling, quality of service (QoS), simulation, validation, wireless sensor networks (WSNs).**

I. Introduction

MODERN complex embedded systems are composed of heterogeneous distributed processing units, electrical and/or electromechanical actuators, and diverse sensing devices. The wiring of such architectures poses further problems in confined space conditions and situations with scattered mounting places. Adding functionality may lead to inflexible and inefficient system design with high maintenance requirements.

Wireless sensor network (WSN) technology may overcome this drawback [1]–[3]. Unfortunately, research in this field is primarily focused on dispersed sensing in unknown environments with subjects such as topology estimation or multihop packet transmission. Reference [4] gives a survey of recent applications, WSN platforms, and research issues in this field. Energy efficiency of sensor nodes is a high priority in the design, triggering numerous research activities toward energy-efficient middleware, protocols, and hardware [3]. An appropriate middleware can increase the reliability and the robustness of the WSN [5]. For industrial embedded products in the aerospace and automobile sectors, the situation is quite different—here, the network topology is usually preplanned and well known. However, WSN technologies such as independent sensor nodes and energy management and harvesting can be exploited. During such a technology change, it is not obvious how the requirements on reliability, latency, etc., of the applications running on it can be assured wirelessly. On the other hand, these nonfunctional properties are of much higher importance for real-time embedded systems than for standard WSNs.

WSNs have numerous design parameters, which may have a major impact on operational and nonfunctional properties. Their system design thus benefits greatly from a model-based analysis. Industrial embedded systems such as vehicles or aircraft are characterized by operational phases (modes) with significantly varying system load and environmental factors. Simulation studies of WSNs are often carried out under simplified worst case assumptions [6] and in unrealistically simple scenarios with homogenous traffic requirements. Those analyses results differ significantly from real scenarios with respect to data traffic, energy consumption, and lifetime. In addition to that, it is not possible to analyze transient behavior at mode changes, which is much harder than a worst case deterministic evaluation simply showing that there is no overload in a stationary environment.

Thus, model-based analysis is used to estimate the system performance to validate quality of service (QoS). QoS is a nonfunctional requirement to the network, which is required by an application. In a heterogeneous network with various applications, it is difficult to guarantee a specific QoS.

There are several types of QoS parameters in WSNs. An overview of these parameters has been given by Chen and Varshney [7]. According to [8], the most fundamental QoS parameters are throughput, delay, jitter, and packet loss rate. Standard network simulators, which have been extended to simulate WSN, include the open source simulators ns-2, OMNeT++, and the commercial OPNET, which are based on discrete event simulation. A comparison of some of the main tools is given in [9]. Other tools specifically address WSN, such as was done in [10] for heterogeneous WSN, using TOSSIM [11] and EmSim/EmStar [12] for the underlying low-level simulation and

Manuscript received July 20, 2013; revised December 24, 2013, May 12, 2014, and August 1, 2014; accepted September 13, 2014. This work was supported by the German Federal Ministry of Economic Affairs and Energy under Grant FKZ:20K1306D.

S. Jäger, R. Maschotta, and A. Zimmermann are with the System and Software Engineering Group, Technische Universität Ilmenau, 98693 Ilmenau, Germany (e-mail: http://sse.tu-ilmenau.de).

T. Jungebloud is with the System and Software Engineering Group, Technische Universität Ilmenau, 98693 Ilmenau, Germany, and also with Mission Level Design GmbH, 99310 Arnstadt, Germany.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSYST.2014.2359690

emulation of sensor nodes and servers. A simulation environment based on MATLAB is presented in [13], including graphical display of sensor nodes in a graphical user interface and the possibility to exchange data between sensor nodes and model.

Even with libraries for application areas such as WSNs, the implementation of a tool environment aimed at system designers who may not be tool specialists is a challenging task. The work presented here is based on a framework for the implementation of such *simulation-based tools* developed recently [14]. We recall its architecture and show its instantiation for the specific area of industrial wireless communication networks.

Apart from the software environment, a domain-specific model generic enough to capture different setups of industrial embedded WSNs is necessary. This paper presents detailed modeling and simulation-based approach for industrial WSN design, using an aircraft cabin environment setting as the specific design task. Time-dependent environmental changes are taken into account, thus leading to a significantly improved prediction of energy consumption, lifetime of sensor nodes, data traffic, and error rates, particularly under changing environmental and operational aspects.

In contrast to our approach, the authors of [15] proposed numerous independent models for different hardware, including, among others, energy models for processor, transceivers, sensors, and nodes. To check requirements for a whole system, however, it is necessary to combine all submodels in one model. This leads to more realistic results because they are closely connected to each other in reality. For instance, if the sensor nodes generate a high amount of traffic on the network, the lifetime reduces because of energy restrictions. On the other hand, if the battery runs low, the sensor node could fall into an energy-saving mode, which produces a smaller amount of traffic. Thus, there is an obvious design tradeoff between energy consumption and network traffic. In this paper, both models are integrated in one single model and jointly analyzed.

A major issue in model-based systems engineering is validation—the results of a prediction can only be trusted if simulation results are successfully compared with a real system. As this is of course not possible for a planned system, smaller test beds or prototypes of typical setups need to be used instead. This is, however, not a trivial task for embedded systems and thus not often done. The authors of [16], for instance, validate QoS parameters for a biomedical sensor network modeled with UPPAAL, a tool for modeling and validating of real-time systems, which are modeled as networks of timed automata [17]. However, the model checking results are only compared with a simulation. The authors of [18] pointed out the importance of validation of WSN implementations and proposed a tool and device for functional checks of an installed setup. This, however, is done to ensure that the system is running correctly as an initial test, and thus termed *deployment time validation*. As we aim at supporting model and simulation validation, this would be too late for our purposes, and we need performance-related properties instead of qualitative checks. Similarly, a *runtime assurance* is proposed in [19], to validate the operation of a WSN in case of changes in operational conditions. A "reliability checking flow," including modeling, simulation, and validation, is sketched in [20]. In contrast to these approaches, our tool supports design time validation by tightly integrating model and prototype: 1) configuration management used both for simulation model and actual WSN setup to avoid errors in transferring design decisions from model to prototype; 2) measuring model by simulation and execution of example WSN applications in a hardware setup; and 3) integrated comparison of resulting data to compare performance values for a validation of model and prototype. This helps in two ways: we can check if the model correctly describes the actual system and—if the model is interpreted as an executable specification of the system—if the real system follows this specification. In the application case study presented in this paper, it actually turned out that the small remaining difference between modeled and real behavior was due to a firmware implementation that differed from the specified standard. This seems to be the case in other WSN scenarios as well, as the analysis in [21] shows for OMNeT++ simulations of IEEE 802.15.4 models. The accuracy of an ns-2 radio link simulation model is studied in [22], including a comprehensive overview of earlier work in this direction.

Another approach uses *hardware in the loop* for the validation of models. Hence, model results are compared with output of an existing hardware to validate functionality of the hardware [23]. In [24], a platform-independent model is used to generate source code for different hardware platforms. In contrast, in this paper, a hardware interface is used to deploy the configuration of various parameters to an existing hardware and compare the results to validate the model.

This extended paper of the Systems Conference 2013 [25] is structured as follows. In addition to the extended introduction and related work, the original paper was extended with an architectural description of a generic framework for simulation-based tools, which can be found in Section II. An architecture model of industrial embedded WSN is given afterward in Section III. In addition to the conference paper, this paper includes a more detailed description of our abstract sensor node, including the new energy components. Like in the conference paper, an avionic application example is presented together with simulation results in Section IV. The added Section V presents a comparison with results from the literature and a recently implemented hardware prototype of an avionic WSN. Due to space limitations, we present only selected results of actual measurements and discuss them in comparison with simulation results, before this paper ends with concluding remarks.

II. Framework for Simulation-Based Tools

In the early phases of a project, model-based engineering helps to make design decisions for a later product. After the conceptual phase, however, the models are often neglected and not used later on.

In the later phases of a project, when there is already a running network in a product, there is no need to have a model to make design decisions for the network itself. In this phase, we have to solve operational issues. An example is the customization of a product depending on a specific customer's needs. A model to check a configuration of a system would be helpful then. At this time in the project, the users are often no system nor modeling experts, but they have to check

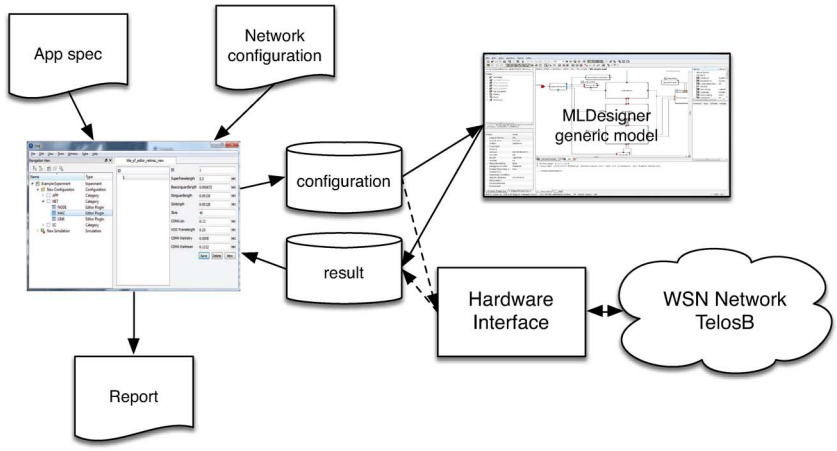

Fig. 1. General tool structure with artifacts and flow of information with additional hardware validation option.

if the customized product will work within the limits of its specification. Here, we can reuse the models that were designed in the concept phase to reduce costs, time, and risk. However, because the later users are often no modeling experts, parameter changes or model adaptations are hard to do or may lead to errors and thus wrong results. Moreover, parameters of the model may be spread over a complex model hierarchy.

To solve this issue, it is important to hide the complexity of the simulation tool and the system model from this user group. To assure that the simulation returns correct results, an application that gives access to the parameters and checks their complex dependencies is needed. This program should also control the simulation and analyze and present the results.

Fig. 1 presents the general structure of a tool that we propose for such a situation. It can be integrated in an engineering design process for validating configurations.

In our case, it consists of three components: a simulation-based application, a database, and the simulation tool with the system model. For validation purposes, an optional hardware interface to the system could be used. The following sections describe the components of our general system structure that we developed to integrate a simulation-based validation of a system configuration into an engineering design process.

A. Simulation-Based Application

The left side in Fig. 1 depicts the simulation-based application itself. This application is the gateway to the system model and can change every parameter of it. The main aim is to hide the simulation logic and the functionality of the simulation tool. It receives both the configuration of the network model and the specification of the applications that run on the network.

Our simulation-based application is based on a framework, which was developed earlier by the authors [14].

For easy extension and customization of the application, we used a component-based approach with a plug-in architecture. New functionality such as new methods for analysis of simulation results can be added with plug-ins. Currently, the application framework has components concerning parameter import, parameter manipulation, simulation control, and result analysis.

Performance parameters and errors are saved to a database during simulation time for later analysis, which is also supported by our simulation-based application. For the later result analysis, there are different kinds of configurable graphs for performance and QoS analysis. An example is shown in Fig. 2, comparing the end-to-end delay of the network for two different applications.

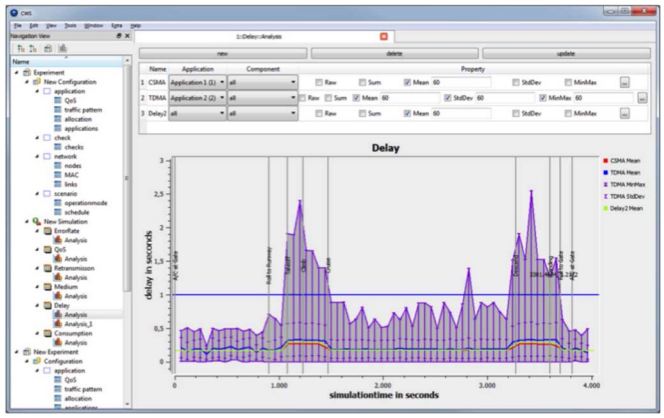

Fig. 2. Screenshot of the simulation-based application with a customized analysis view.

It is possible to generate reports with configuration and all analysis information for documentation and reporting issues.

More details of the software architecture for simulation-based applications are contained in [14].

B. Database

The database serves as data storage and point of data exchange between simulation-based application and simulation tool with the system model. It is implemented with Open Database Connectivity (ODBC) and uses the same interfaces in the simulation-based application and the simulation tool with the system model. By using ODBC as database abstraction layer for simple selection and insertion tasks, it is possible to interface different database back ends such as Oracle SQL, MySQL, or others. For our application, we use the SQLite Database. The main advantage is its simplicity and memory efficiency, and there is no need to install a full-size database server. Our application uses the database to store the model configuration for the simulation and to save the simulation results.

C. Simulation Tool

The model was implemented in the multidomain simulation tool MLDesigner [26]–[28]. It allows hierarchical models by using block diagrams. For our model, we used the discrete event domain for the network and the lifetime model and finite state machines for the energy consumption description. To achieve high scalability of a model, i.e., independence of the model size and structure from the number of similar objects in the system, MLDesigner allows generating instances of a template during runtime of the simulation, termed as *dynamic instances*. It is not only a network simulator like NS-2 or OMNeTT++, but it is more generic and simplifies system environment modeling known as the *mission level design* approach. Libraries of modeling elements for a variety of application areas simplify modeling of complex systems [29]. It has been applied to ZigBee networks earlier [30], among many other application fields.

D. Hardware Interface

For validation purposes, it is useful to provide a possibility to deploy the configuration from the configuration database to a real system environment directly. This greatly simplifies starting and controlling the system. A hardware interface for our tool has been implemented for this task recently. In our example setup, it is connected to a sensor node via an RS232 interface. An interface similar to the simulation control is used on the software side. It implements access to the configuration database and deploys configuration setups to the system. As a result, the configurations of simulation model and hardware target are identical, avoiding errors in a translation. During a run of the actual system, the schedule used for the model is also used to control parameters of the hardware. Moreover, it is possible to import results measured from the prototype into the result database with the hardware interface, and thus to compare them with simulation results for a direct validation. Fig. 1 visualizes this validation tool chain.

III. Model of Industrial Embedded Sensor Networks

This section describes a model of an embedded WSN and its applications for use in an aircraft environment. The topmost model consists of two components, namely, the environment and the network itself. The environment generates input for the network model. It consists of three parts: the schedule, the traffic generator, and the energy generator.

The *scheduler* serves as the timetable for the simulation. It defines a sequence of *operation phases* with their durations, depending on the configuration in the database. At the start of the simulation, it calculates the occurrences of phase switching events and enqueues them in the event scheduler.

The *traffic generator*, which controls the traffic load generated by sensor nodes and their applications, uses the operation phases to determinate the current amount of traffic each sensor node produces. It is possible to assign different traffic patterns to each node depending on its application and current operation phase. By using dynamic instances in the traffic generator for a sensor node, it is possible to run more than one application on a sensor node, and the resulting traffic is a mixed pattern. Currently, our model supports four different kinds of patterns: periodic, a probabilistic one with an exponential interarrival time, a single event pattern, and a zero-traffic pattern.

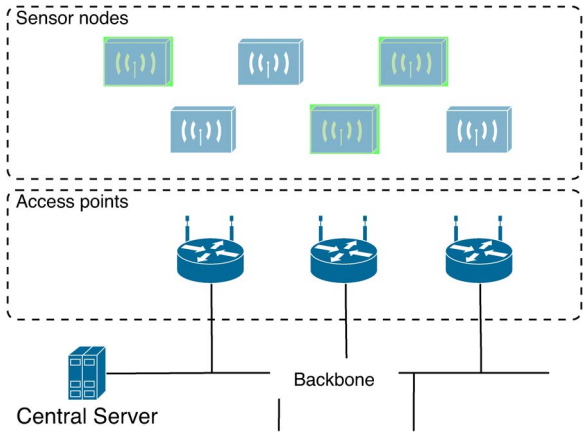

Fig. 3. Top-level view of the WSN model.

The third part describes *energy consumption and generation*, which calculates the amount of energy a sensor node can obtain depending on its physical position. This can be influenced by the current phase as well.

The network model is, in our avionic application, a simplified WSN with a centralized control server. Its top level is depicted in Fig. 3 and consists of a central control server, the backbone network, a set (cloud) of access points (AP), the shared medium, and a set of sensor nodes (SN). The cloud of APs and SNs are designed as templates in the model and can be configured and instantiated during runtime of the simulation. Each instance is connected to the others by an event channel, which transfers packets to others. It thus behaves like a shared medium as in reality and can produce collisions and errors. In addition to that, we can add interferers to disturb the communication and capture channel failures. For the communication, three different kinds of protocols have been implemented so far: a stochastic one, a deterministic one, and a hybrid one that combines the characteristics of the two others. The hybrid protocol is an abstract implementation of the IEEE 802.15.4 Standard. They are modeled at different levels of abstraction.

The access points are connected to the control server via a wired backbone network. The routes over which packets are transported are defined before the simulation starts. The control server stores the routes for each end-to-end connection and statically routes the packets.

The sensor node is one of the major system components and will be described in detail in Section III-A.

The driving force of our simulation is *applications*, which are described by an application specification each. It contains application restrictions such as maximum end-to-end delay and maximum packet loss. To assign individual properties to an application, the generated traffic can be fed into each sensor node individually. Each application may have different traffic patterns for each phase in the schedule. Operational phases and their mission schedules can be freely defined in our model and tool. At the moment, the generator can produce stochastic, periodic, and single-packet data traffic.

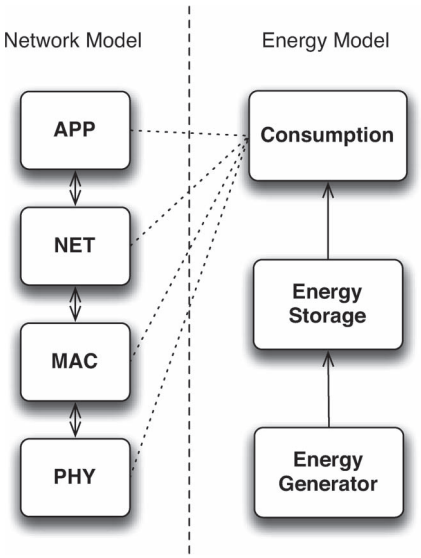

Fig. 4. Block chart of the abstract sensor node.

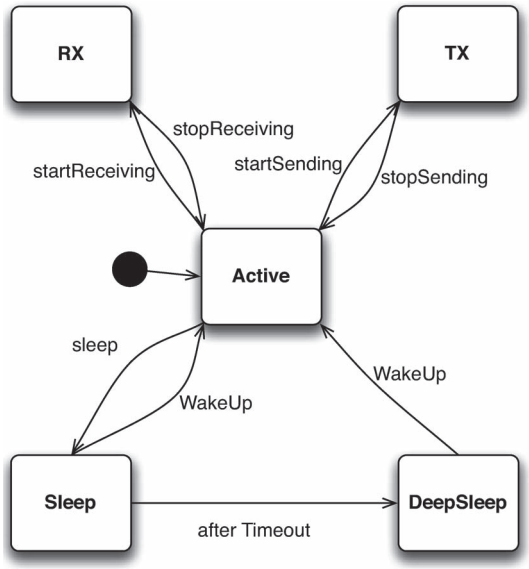

Fig. 5. State machine for energy consumption of the sensor network.

Performance parameters and errors are saved in a database during simulation and can be analyzed afterward.

A. Abstract Sensor Node

The whole network model was built in a top-down modeling process starting with the top level (see Fig. 3) and its refinement down to the medium access control (MAC) layer of the network. The physical layer is not considered yet and abstracted by a bit error rate, which, at the moment, simply depends on the distance between communication partners.

Each sensor node combines the network model for transporting data and an energy model, which calculates consumption and generation of energy.

Fig. 4 shows a block chart of the abstract sensor node with the two models. They are connected by trigger signals. The following subsections will describe the two models.

1) Network Model: The network model of an abstract sensor node is based on a reduced Open Systems Interconnection layer model for embedded systems, which consists of application, network, medium access control, and physical layer (cf., left side in Fig. 4). By using a defined interface for each layer, we can easily replace them with a different level of abstraction. With this "plug-and-play" approach, it is even possible to change the protocol of the network. Each layer is connected to the consumption component and triggers it for resulting calculations.

2) Energy Model: Each sensor node has an associated energy consumption function in the model, which calculates the consumption according to the current activity of the sensor node. Based on the energy consumption function and the lifetime model, the energy-induced reliability (lifetime) of the system is determined during a simulation run.

The energy consumption model is based on an event trigger mechanism. The energy model consists of three components: energy consumption, energy storage, and energy harvesting. A suitable energy consumption model is the basis for developing and evaluating a power management scheme in the WSN. Consumption depends on the operating mode of a device such as whether it utilizes sleep modes to reduce consumption and how frequently it transmits and receives data. The internal representation is a simple state machine, which is the abstract model for energy consumption. The energy consumption is calculated taking into account all the components of the network model. The corresponding state machine diagram is described in Fig. 5.

The state machine diagram specifies in what mode of operation the sensor node consumes which amount of energy and how state transitions take place. It has five states: "Active," "TX," "RX," "Sleep," and "DeepSleep."

The current state of the state machine depends on the activity of the sensor node itself. At the beginning of the simulation, the machine starts in Active state. If there is no activity in the network model of the sensor node, it switches into Sleep state. If there are no events for an (specifiable) amount time, the node switches into DeepSleep to save energy.

If the application layer starts to generate data, because a time trigger occurs, the state machine switches to "Wake up" and goes back to active state. The same happens if the network model of the sensor node receives incoming data. After a wake up time, it switches to TX or RX state. In TX or RX, it sends or receives data to or from the shared medium and switches into sleep mode afterward. Sometimes, the sensor node wakes up and performs tasks such as synchronization on a beacon or listening on the medium for incoming traffic, which do not lead to a communication.

Following [31], an energy storage model should describe capacity, energy density, internal resistance, average leakage current, and minimum charging current. This component can be equipped with different types of energy storage modules such as supercapacitors and rechargeable batteries. The energy harvester model may describe the different technology types of actual harvesting processes by adjusting the characteristics of the parameters of the module. The production of energy is affected by the power generation parameters of the environment model.

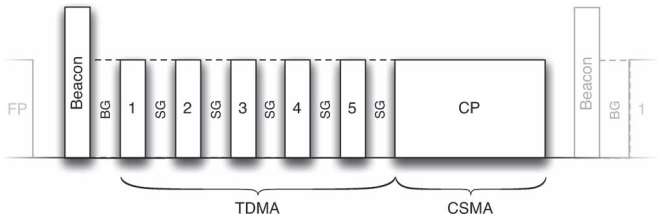

Fig. 6. Network protocol: Access point frame structure.

TABLE I
PARAMETERS FOR THE MAC LAYER

parameter	value
superframe length	16000 byte
ap frame length	4000 byte
beacon length	2 byte
beaconguard length	2 byte
slot length	40 byte
slot guard	40 byte
slot count	40
csma length	200 byte
csma access mean	0.00256 seconds
csma wait retry	0.00256 seconds
byte time	0.000032 seconds/byte

IV. EXAMPLE NETWORK

As an example, we created a small network running two applications. In the next sections, we will describe the structure of the actual network, the environment, and its applications. The network will be analyzed with three different scenarios afterward.

A. Structure

The example model is based on the previously described model architecture. It contains a reduced amount of components to validate the accuracy of the model. In Fig. 3, the logical structure is shown. For our simplified model, we use four dynamic instances for access points (AP) and 30 dynamic instances for sensor nodes (SN). The four APs are connected to the control server via a backbone network with a daisy chain topology. The delay of data because of the daisy chain depends on the position of the AP in the chain. This leads to an additional delay on the backbone network for end-to-end transmissions.

For our network, we use a hybrid protocol, which provides deterministic and stochastic medium access. Fig. 6 sketches the structure of an access point frame in the superframe of the hybrid protocol.

A frame in the protocol begins with a beacon, which is a start signal for the frame, followed by a beacon guard to reduce the chance of collision. Then, the deterministic part starts. Every sensor node, which is assigned to the deterministic part, has its own uplink and downlink slots for bidirectional communication. The slots are separated by guards. The stochastic part begins after the last slot guard. Sensor nodes assigned to this part access the medium via carrier sense multiple access/collision avoidance. They have concurrent access to the shared medium. Table I shows the required parameters for the MAC layer in our example.

TABLE II
SPECIFICATION OF APPLICATION PARAMETERS

	Background load	Application
characteristics	periodic	periodic
time critical	no	yes
deadline	-	1 sec
number of SN	20	10

TABLE III
SCHEDULE FOR A FLIGHT OF 4000 s WITH DIFFERENT PHASES AND CORRESPONDING TRAFFIC PATTERN FOR BACKGROUND LOAD AND APPLICATION

phase	start time	application (period)	background (period)
parking	1 sec	10.00 sec	1.00 sec
taxi	901 sec	5.00 sec	1.00 sec
takeoff	1081 sec	1.00 sec	0.75 sec
climb	1231 sec	1.00 sec	0.75 sec
cruise	1471 sec	10.00 sec	1.00 sec
descend	3271 sec	2.00 sec	0.75 sec
landing	3600 sec	1.00 sec	0.75 sec
taxi	3700 sec	5.00 sec	1.00 sec
parking	3811 sec	10.00 sec	1.00 sec

TABLE IV
CONSUMPTION OF A SENSOR NODE IN THE STATES

state	consumption
Active	1.8000 mA
TX	19.5000 mA
RX	23.8000 mA
Sleep	0.0210 mA
DeepSleep	0.0005 mA

B. Environment

The environment controls traffic generation and energy production models of the sensor nodes. For our example simulation, we use one flight of an aircraft. The schedule describes different phases in the flight of an aircraft, such as taxing, cruising, and so on (see Table III).

In our example, the sensor nodes are powered by a battery and will not be recharged by energy harvesting. A future extension of our model will describe harvesting based on operational phases and sensor locations.

C. Applications

For our example, we used two applications with different traffic characteristics. A detailed specification of both applications, including end-to-end delay requirements, is shown in Table II.

The background load is modeled as an application in our network configuration. It produces data traffic in the particular phase, as shown in Table III. The amount of traffic varies between noncritical and critical flight phases. It serves as a background load for our simulation and simulates the already existing applications in the embedded wireless network, before a new assumed real-time application communication is analyzed.

The example captures the behavior of a typical logging application, which measures the temperature of a sensor in a defined interval. The application has additional requirements with

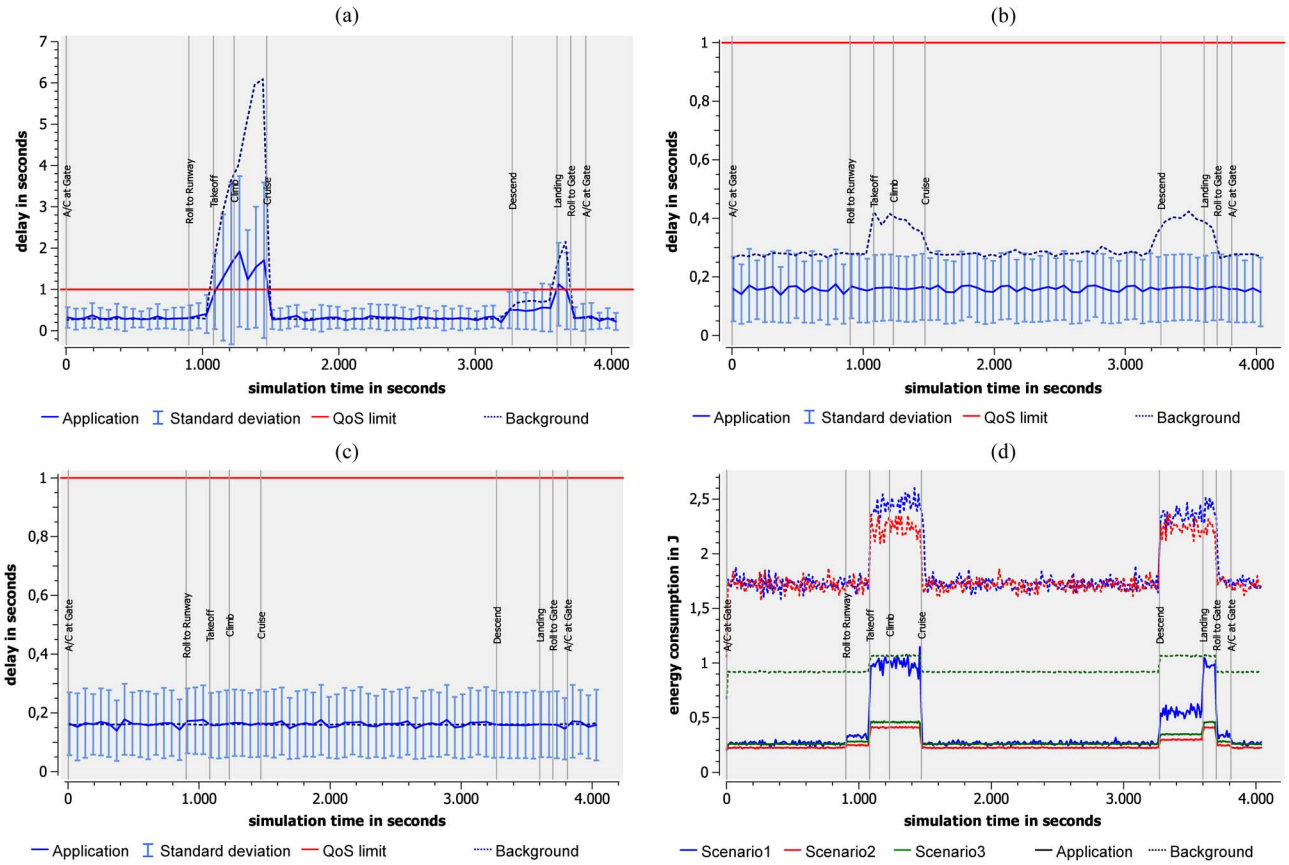

Fig. 7. (a)–(c) Mean delay of background load and application with the standard deviation of the application for the different scenarios. The horizontal red line shows the limit for the maximal delay of the application. The averaging period is 60 s. (d) Energy consumption of the applications for all scenarios. (a) Delay scenario 1 (CSMA/CSMA). (b) Delay scenario 2 (TDMA/CSMA). (c) Delay scenario 3 (TDMA/TDMA). (d) Energy consumption.

regard to end-to-end delay and number of lost packets. Table III describes traffic patterns for the particular flight phases.

There are critical phases in which significantly more data have to be acquired and transferred. Typically, this may happen during takeoff and landing. A simulation of an emergency scenario such as fire would be another example. The data size of one packet is 36 B. However, only the sample frequency and not the amount of data per sample is changed.

D. Scenarios

To analyze the impact of the protocol on timing requirements, energy consumption, and lifetime, three different experiments are considered. In the first scenario, both applications access the medium via carrier sense multiple access (CSMA). This means that they can block the other application while sending. The second scenario uses a hybrid medium access protocol, in which one application uses time-division multiple access (TDMA), and the other uses CSMA. Finally, both applications are communicating via TDMA in the third setup.

E. Energy Consumption

Energy consumption of the sensor node depends on the states of the state machine and how long it remains in that state. In each state, a sensor node uses a state-dependent amount of energy (see Table IV).

The parameters for the consumption are extracted from a data sheet of the popular wireless sensor platform TelosB [32] and a publication comparing different wireless sensor platforms with each other [33]. This hardware is used for model validation experiments, which are presented in Section V-B1.

F. Results

In the following section, the results of our simulation in the three scenarios are described.

1) Scenario 1: Fig. 7(a) shows the development of packet delay for background load and applications during a simulation for the first scenario.

As a result, both applications run in the same part of the protocol and influence each other while trying to access the medium with a concurrent protocol. The mean delay corresponds to the current load of the network.

The application has a time restriction for its maximum end-to-end delay (represented by the red line) of 1 s. The delay fails to meet the requirement three times (takeoff, climb, landing) when the load is high. The peak mean delay is about two times higher than the limit. With these results, it is obvious that running both applications with CSMA will not fulfill their requirements.

For scenario 1, the overall consumption for the background and the application is presented in Fig. 7(d). The averaging period is 10 s.

The consumption corresponds to the current load of the network model. It is directly connected to the amount of data the sensor nodes have to send. The peak consumption of the background load is around 2.5 J (0.125 J for an SN). The lowest is around 1.7 J (0.085 J for an SN). For the application, the peak consumption is around 1 J (0.1 J for an SN). The lowest is around 0.25 J (0.025 J for an SN).

The higher consumption of a sensor node in the background follows from slightly more traffic generation of the sensor nodes at peak consumption of the background.

2) Scenario 2: In scenario 2, the TDMA part of the protocol is used for the time-critical communication. The background load stays in the CSMA part. The resulting behavior of the mean delay is shown in Fig. 7(b).

With the results, it is easy to see that the delays of both applications are now independent. They do not influence each other unlike in the previous scenario. The mean delay of the background is still high in phases with high traffic, but the peak delay is much smaller than in the previous scenario. The reason is that the number of sensor nodes in the CSMA part drops from 30 to 20. Thus, only 20 sensor nodes are accessing the medium concurrently.

The other ten sensor nodes only have one time slot in the TDMA part of the protocol. The maximum delay no longer corresponds to the network load but to the length of the superframe and data generation of the sensor node, which is described by MAC configuration and traffic pattern. The maximum delay of the application is around 0.15 s, and the maximum deviation is 0.15. In the worst case, the delay of a packet of the application is 0.3 s.

The energy consumption of the applications for scenario 2 is presented in Fig. 7(d). The peak consumption of the application drops from 1.0 J in scenario 1 to around 0.45 J. That is because of the concurrent access of the sensor nodes on the shared medium, where a sensor node needs more attempts to send a packet. The value of the lowest consumption stays at 0.25 J because, in this phase of the simulation, the traffic is very low. Operations such as receiving beacons or the activity of the microcontroller unit (MCU) need more energy than sending payload packets.

The peak consumption of the background load drops from 2.5 J to around 2.3. Because the application uses TDMA for medium access, the background is not influenced anymore. The value of the lowest consumption is also the same as in scenario 1 because of the low influence of the application in this part.

Thus, the model-based simulation analysis shows that the requirements can be fulfilled with a hybrid approach for the applications.

3) Scenario 3: In the last scenario, we assign both applications to the TDMA part of the protocol. The results are depicted in Fig. 7(c).

Now, each of the 30 sensor nodes communicates in its own time slot, and they will not influence each other anymore. As in scenario 2, the mean delays of the applications are only influenced by superframe length and data generation of each node.

Both background load and application have the same value for the mean delay. It is around 0.15 s. The standard deviation for the application is 0.15.

It is important to choose the parameters of the MAC layer depending on the requirements of applications. For example, a critical application, which needs a value every 0.5 s, cannot run with an MAC with a superframe length of 2 s.

The energy consumption for scenario 3 is presented in Fig. 7(d). The consumption of the application stays the same (the peak value is 0.45 J, and the lowest is 0.25 J). The consumption of the background now drops to 1.05 J in peak and around 0.9 J for the lowest value. Adjusted according to the number of sensor nodes and traffic, a sensor node consumes the same amount of energy in background and application.

4) Discussion: As a result, we can check requirements of the applications with a model-based approach. By simulating an application in its different phases, the model delivers more accurate results than a simplifying worst case simulation. Transient effects at mode changes can be analyzed. By using a concurrent protocol, the current consumption of a sensor node is connected to the amount of traffic in the network. As an example result to be demonstrated, in cases of very low data traffic, a stochastic protocol can be efficient with regard to energy consumption and latency. On the other hand, in a scenario with very high traffic and lots of sensor nodes, a deterministic protocol is more efficient.

V. Validation

This section describes the validation of the model by comparing the simulated behavior of our model with results from existing publications as a benchmark and with measured parameters of a real WSN. For this purpose, the hardware interface of the simulation-based application is used, which has been described before.

A. Comparison With Literature

First, we validate the output of our model by comparing the results with related publications.

For this purpose, we use the configuration of the second scenario (see Section IV-D) and make some small changes.

We simulate CSMA and TDMA each with 15 nodes and set the message rate of each node from 2^{-2} to 2^{5} message/s. The observed parameters include end-to-end delay and throughput.

Fig. 8(a) depicts the end-to-end delay. In low-data-rate cases, the sensors can send all their packets. When the data rate is increased, the packets are enqueued in the sending queue, and the delay exponentially increases for TDMA and CSMA. The same characteristic curve can be seen in [34] and [35].

Fig. 8(b) presents the throughput. It increases with the data rate and reaches a maximum. Because there is no concurrent access between TDMA nodes, the throughput for TDMA is much higher than for CSMA. Moreover, our results show the characteristic decrease of CSMA throughput above the optimal data rate that has been identified in the literature [34]–[36].

B. Hardware Validation

1) Prototype Hardware Environment: Our hardware target is based on the well-known wireless sensor platform TelosB, which was developed at the University of California, Berkeley.

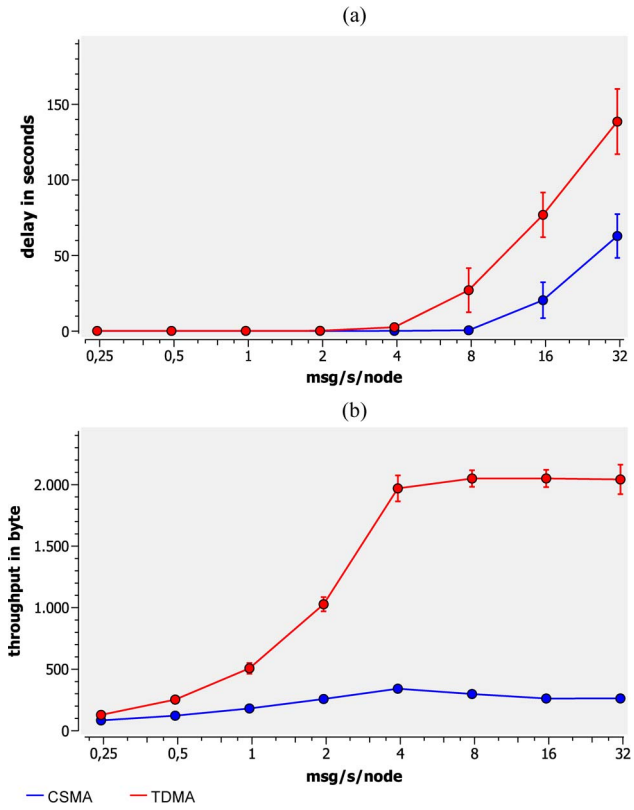

Fig. 8. Delay and throughput in relation to message rate per node. (a) Delay for CSMA and TDMA. (b) Throughput for CSMA and TDMA.

It is based on an MSP430 MCU and uses a CC2420 chip as transceiver. Further information can be found in the data sheet [32]. The *motes* serve as remote sensor nodes and access points.

The sensor node is running a customized version of TinyOS 2.x, an open source operation system for low-powered devices such as wireless sensor nodes [37]. For our purposes, we created an application layer that can be configured to generate different amounts of packet traffic depending on the current traffic pattern. We also created a new collision avoidance layer in the rfxlink network stack that realizes the hybrid protocol of our simulation model, which we described in Section IV-A. All parameters that can be set in the model can be also set in the real hardware.

The hardware interface of the simulation-based application is connected to a TelosB mote, which serves as access point, via an RS232 interface.

2) Experiments: The hardware interface runs a fixed sequence of steps to obtain repeatable results. The workflow starts with a reset of the whole network to ensure that all sensor nodes are in the same state. This reset deletes all data of a previous run.

After the reset, the hardware interface tries to find all available network nodes by using discovery messages, which the sensor nodes respond to. After the discovery step, the hardware interface maps the available sensor nodes to the nodes in the configuration and pushes the individual configuration of each component to its corresponding node. This contains the sensor node configuration, the configuration of the medium access control, and the traffic pattern for the application, which is running on the node.

The network needs a global time to measure performance parameters of the network such as end-to-end delays. Therefore, a time synchronization phase is conducted. The *flooding time sync protocol* (see [38]), which is part of TinyOS, is used to synchronize the clocks of sensor nodes and access points. The protocol provides an accuracy of the global time in the range of microseconds, which is entirely sufficient for our purpose.

After the configuration and synchronization phases have finished, the experiment starts. The hardware interface obtains the schedule of the simulation from the configuration database and runs it on the hardware. To avoid additional traffic on the network, which might otherwise affect the measured results, the current phase of the schedule is distributed by the beacon as payload. During the experiment, the hardware interface collects performance data of the network and saves it into the result database.

The hardware interface runs the experiments multiple times to reduce noise and disturbance from interferers, which also use the 2.4-GHz industrial, scientific, and medical radio spectrum. Each repetitive run is begun with a reset. Finally, the results can be analyzed and compared with each other and with the simulation results in the simulation-based application.

In the following, the experiments that were used to validate the WSN model are outlined. To validate the model, each level of detail has to be checked against the real hardware. This can be done by variations of parameters of the configuration and comparing the output of the simulation and the hardware target.

There are many different parameters of the network that can be observed, including energy consumption, packet loss, collisions, and throughput. We restrict ourselves to analyzing the end-to-end delay here.

The parameters for the validation follow the parameters described in Section IV. In the case of parameters that cannot transferred to the hardware in principle, such as the physical parameters of the hardware, the model parameters are adapted to the setup.

The original model, for instance, used other parameters for modulation and frequency. For instance, the *symbol rate* parameter of the model, which is strongly influenced by modulation and frequency, has been set to the value 0.000016 s per symbol for TelosB.

a) Experiment 1: The aim of the first validation experiment is to check the behavior of the TDMA part of the medium access control. For that experiment, a simple network topology is used. It consists of one access point and two sensor nodes, with each sensor node running a different application. Both use TDMA to transmit data to the access point. The applications use two different traffic patterns. For the first application, the period of the traffic pattern, i.e., when it generates data, is a multiple of the length of the superframe. It is 1024 ms, whereas the superframe length is 512 ms. In the traffic pattern of the second application, the period slightly differs from the multiple period of the superframe. The period time is 1000 ms here. The sensor nodes use successive slots of the medium access control to see possible effects of interference. The remaining part of the configuration, including MAC parameters, remains the same as described before.

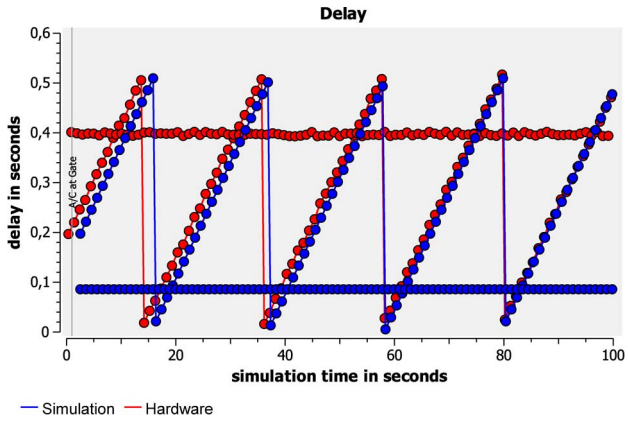

Fig. 9. Results of the simulation and the hardware for TDMA.

Fig. 9 shows the resulting curves of the experiment for the simulation target in blue and the hardware setup in red.

The timespan was reduced to 100 s to ease understanding of the results.

The sensor node with the first application produces a straight curve (lower line) in the output of the simulation and the hardware (upper line). Comparing the two curves, the output of the hardware fluctuates around an average delay, whereas that of the simulation always has the same delay without any noise.

The average delays of both curves are not the same, because of the start of the traffic generation, which is randomly chosen on both sides. The possible values can be inside the length of a superframe, under 512 ms.

The curve for the second application looks completely different from the one of the first application. The slightly different period time value leads to a sawtooth-like wave. The time when the packet is generated from superframe to superframe sweeps over the whole superframe length. The output of the simulation and the hardware only differ in small fluctuations of delay on the hardware.

A big advantage of TDMA w.r.t. predictability is that sensor nodes cannot disturb each other because they use different slots to transmit packets to the access point. The results of simulation and hardware also show this behavior, as there are no influences between the curves of both applications either in the model or in the hardware.

The results of the first experiment show that the model behaves similar to the hardware network concerning the delay of a packet, which was transmitted via the TDMA part of the medium access control.

b) Experiment 2: The second experiment checks the behavior of the CSMA part of the medium access control.

Different from the first experiment, the number of network components is increased. There are now five sensor nodes in the network. By using the CSMA area of the network, the nodes can disturb each other while trying to send packets to the access point.

The network runs two different applications just like in the first experiment. The traffic patterns of both applications are set as described in Section IV-C.

Fig. 10(a) shows the results over the full length of the run. To reduce probabilistic effects such as noise and sporadic

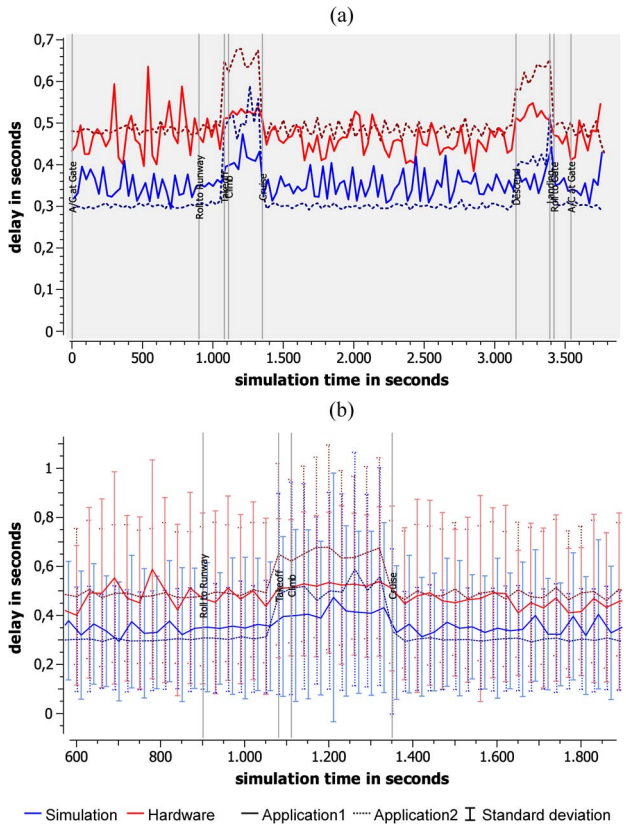

Fig. 10. Results of the end-to-end delay for CSMA transmission. The averaging period is 30 s. (a) Delay full. (b) Delay detail.

disturbances, the hardware experiment is run 15 times. The blue curves represent the output of the simulation, whereas the red ones represent the output of the hardware. The background application is represented by the dashed line, and the application is represented by the solid one.

The curves show the same behavior as in the simulation of the whole network. The smaller amplitude is caused by the reduced number of sensor nodes in the validation experiment.

The curve of the background starts smoother than the one for the application. This is because of the difference in the number of packets that are used to calculate the mean value. This value is around 0.48 s for the hardware and 0.3 s for the model. In the later phase of the schedule, such as takeoff and climb of the aircraft, the delay increases because of more traffic. Here, the mean delay for the background traffic jumps up to 0.68 for the hardware and 0.58 for the simulation. Influenced by the background traffic, the delay of the application also increases. After switching into the cruise phase, both curves fall back to the normal delay after clearing their sending queues. During descend and landing phase, the mean delay increases again. At the end of the schedule, the value of the mean delay is back to where it started.

Fig. 10(b) shows a part of the experimental results where the traffic pattern switches from low to high traffic and back. The standard deviation is shown as well.

The standard deviation of the simulation follows the curve of the mean delay. While the traffic is low, the deviation is small and vice versa. The hardware behavior differs slightly.

Its standard deviation for the application decreases while the traffic is high. Over the entire term, the standard deviation of the hardware is higher than the standard deviation of the simulation.

The curve for the mean delay of the hardware is around 0.1–0.2 s higher than the curve of the model. This effect is attributed to a delay in the access of the medium in the hardware, which is not modeled until now.

There is also a difference in the number of packets that can be transferred through the medium. The average number of packets per run that was transmitted by the hardware is 11 991.1, whereas the simulation could transmit 13 422.8 packets at the same time. This fact influences the standard deviation of the curves in phases of high traffic.

For CSMA, the output of the simulation has the same behavior, but there are some small differences in delay and number of transmitted packets.

C. Interpretation of Results

The comparison of the model with related publications shows similar results for TDMA and CSMA transmissions. In addition to that, our detailed comparison with an actual hardware test bed setup validates and proves the quality of the simulation result both for TDMA and CSMA.

Because the model is free of noise and disturbances, the simulation results are smoother than the ones of the hardware and return the expected value for the observed parameter. The result of the hardware converges to the expected value as well, but needs more runs.

The differences in the results of the hardware validation, mostly for the CSMA transmission, are based on the level of abstraction of the model. Not every detail of the hardware has been modeled, for example, the time that a sensor node needs to switch between TX and RX state, delays for the network layer, or effects such as shadowing and reflections. These effects mostly influence the concurrent access of the medium. To get more accurate results for CSMA, the model has to be refined w.r.t. interferences between sensor nodes. Nevertheless, the model allows a sufficient estimation of the end system behavior.

The model allows a faster checking of configurations: The mean runtime of a simulation run is 22 s compared with one run of the hardware experiment taking 63 min.

The main advantage of our approach of deploying only network component configurations is the possibility to evaluate parameter sets significantly faster and autonomous. In contrast to [24], it is not necessary to generate and deploy full firmware binaries to the devices, leading to a much faster validation cycle, including configuration, measurement, and simulation. We point out that the validation is very detailed because of its focus on lower layers (NET, MAC, PHY) of the protocol.

By using the model as a reference during the creation of the hardware target, errors in the implementation of the sensor nodes and the access point were found early. In addition, the location of the error could be estimated out of the model. On the other hand, the hardware target reveals inexactly modeled parts of the model. As a result, the developed hardware system could be now double checked, which will lead to a higher overall system quality.

VI. Conclusion

This paper has presented a design of a highly scalable industrial embedded WSN for validating different system configurations by using a domain-specific simulation-based tool. The tool supports a seamlessly integrated approach to modeling, configuration, performance evaluation, design time validation, and result interpretation.

An avionic system design example has been presented. QoS parameters of an application can be evaluated more accurately: operational phases and their influence on time-dependent system operation are specifically addressed. Simulation shows transient effects, for instance, after mode changes. The application example shows how the model evaluation is used to check if MAC parameters are chosen well or if overload situations may occur. Models of network and energy consumption are tightly connected.

Our approach of deploying configurations has been shown to be a significantly faster way to test several parameter variations. By adding a parallel hardware test bed to our tool, different configurations are autonomously deployed to simulation and hardware and are analyzed with the same methods by the simulation-based tool.

Measurements validate the correctness of simulation and model and even detected a difference between prototype system and expected implementation.

In the future, we plan to refine the model w.r.t. interferences between sensor nodes. Moreover, energy-efficient protocols and energy harvesting components will be added and validated with our hardware setup.

References

[1] J. Zheng and A. Jamalipour, *Wireless Sensor Networks: A Networking Perspective*. Hoboken, NJ, USA: Wiley, 2009.

[2] 3-8322-4760-2J. Beutel, "Design and deployment of wireless networked embedded systems," Ph.D. dissertation, ETH Zürich, Zürich, Switzerland, 2005.

[3] G. Anastasi, M. Conti, M. Di Francesco, and A. Passarella, "Energy conservation in wireless sensor networks: A survey," *Ad Hoc Netw.*, vol. 7, no. 3, pp. 537–568, May 2009.

[4] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Comput. Netw.*, vol. 52, no. 12, pp. 2292–2330, Aug. 2008.

[5] S. Hadim and N. Mohamed, "Middleware for wireless sensor networks: A survey," in *Proc. 1st Int. Conf. Comsware*, 2006, pp. 1–7.

[6] Y. Liu, W. Zhang, and K. Akkaya, "Static worst-case energy and lifetime estimation of wireless sensor networks," *JCSE*, vol. 4, no. 2, pp. 128–152, Jun. 2010.

[7] D. Chen and P. K. Varshney, "QoS support in wireless sensor networks: A survey," in *Proc. Int. Conf. ICWN*, Las Vegas, NV, USA, Jun. 2004, pp. 1–7.

[8] F. Xia, "Qos challenges and opportunities in wireless sensor/actuator networks," *Sensors*, vol. 8, no. 2, pp. 1099–1110, 2008.

[9] M. Jevtić, N. Zogović, and G. Dimić, "Evaluation of wireless sensor network simulators," in *Proc. 17th TELFOR*, Belgrade, Serbia, 2009, pp. 1303–1306.

[10] L. Girod *et al.*, "A system for simulation, emulation, deployment of heterogeneous sensor networks," in *Proc. 2nd ACM Conf. Embedded Netw. Sens. Syst.*, 2004, pp. 201–213.

[11] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and scalable simulation of entire TinyOS applications," in *Proc. 1st Int. Conf. Embedded Netw. Sens. Syst., ser. SenSys '03*, 2003, pp. 126–137.

[12] L. Girod *et al.*, "EmStar: A software environment for developing and deploying wireless sensor networks," in *Prof. USENIX Tech. Conf.*, 2004, pp. 283–296.

[13] A. Cardoso, S. Santos, A. Santos, and P. Gil, "Simulation platform for wireless sensor networks based on the truetime toolbox," in *Proc. 35th Annu. Conf. IEEE IECON/IECON*, 2009, pp. 2115–2120.

[14] R. Maschotta, S. Jäger, T. Jungebloud, and A. Zimmermann, "A framework for agile development of simulation-based system design tools," in *Proc. IEEE Int. SYSCON*, 2013, pp. 861–866.
[15] H.-Y. Zhou, D.-Y. Luo, Y. Gao, and D.-C. Zuo, "Modeling of node energy consumption for wireless sensor networks," *Wireless Sens. Netw.*, vol. 3, no. 1, pp. 18–23, Jan. 2011.
[16] S. Tschirner, L. Xuedong, and W. Yi, "Model-based validation of QoS properties of biomedical sensor networks," in *Proc. 8th ACM Int. Conf. Embedded Softw.*, *ser. EMSOFT '08*, 2008, pp. 69–78.
[17] J. Bengtsson, K. Larsen, F. Larsson, P. Petterson, and W. Yi, "UPPAAL—A tool suite for automatic verification of real-time systems," in *Proc. DIMACS/SYCON Workshop Hybrid Syst. III, Verification Control*, 1996, pp. 232–243.
[18] H. Liu, L. Selavo, and J. Stankovic, "SeeDTV: Deployment-time validation for wireless sensor networks," in *Proc. 4th Workshop Embedded Netw. Sens.*, *ser. EmNets '07*, 2007, pp. 23–27.
[19] Y. Wu *et al.*, in *Proc. 9th ACM/IEEE Int. Conf. Inf. Process. Sens. Netw.*, *ser. IPSN '10*, 2010, pp. 197–208.
[20] Z. Shen, K. L. Man, C.-U. Lei, E. G. Lim, and J. Choi, "Assuring system reliability in wireless sensor networks via verification and validation," in *Proc. ISOCC*, 2012, pp. 285–288.
[21] J. Rousselot *et al.*, "Accurate timeliness simulations for real-time wireless sensor networks," in *Proc. 3rd UKSim EMS*, 2009, pp. 476–481.
[22] G. Wittenburg and J. Schiller, "A quantitative evaluation of the simulation accuracy of wireless sensor networks," in *Proc. 6th GI/ITG KuVS Fachgespräch "Drahtlose Sensornetze"*, 2007, pp. 23–26.
[23] S. Palla, A. Srivastava, and N. Schulz, "Hardware in the loop test for relay model validation," in *Proc. IEEE ESTS*, May 2007, pp. 449–454.
[24] M. M. R. Mozumdar, L. Lavagno, L. Vanzago, and A. Sangiovanni-Vincentelli, "Hilac: A framework for hardware in the loop simulation and multi-platform automatic code generation of WSN applications," in *Proc. Int. SIES*, Jul. 2010, pp. 88–97.
[25] S. Jäger, T. Jungebloud, R. Maschotta, and A. Zimmermann, "Model-based QoS evaluation for embedded wireless sensor networks," in *Proc. IEEE Int. SysCon*, Orlando, FL, USA, Apr. 2013, pp. 195–199.
[26] G. Schorcht *et al.*, "System-level simulation modeling with MLDesigner," in *Proc. 11th ACM/IEEE Int. Symp. MASCOTS*, 2003, pp. 207–212.
[27] A. Agarwal, C.-D. Iskander, R. Shankar, and G. Hamza-Lup, "System-level modeling environment: MLDesigner," in *Proc. 2nd Annu. IEEE SysCon*, Montreal, QC, Canada, Apr. 2008, pp. 1–7.
[28] "Mldesigner," Palo Alto, CA, USA.
[29] T. Baumann, A. Pacholik, and H. Salzwedel, "Performance exploration with MLDesigner using standardised communication interfaces," in *Proc. DATE*, 2007, pp. 1–2.
[30] F. Lohse and V. Zerbe, "Model based performance estimation in zigbee based wireless sensor networks," in *Proc. 27th. Int. Conf. MIEL*, 2010, pp. 307–310.
[31] W. S. Wang, N. Wang, M. Hayes, B. O'Flynn, and C. 'Mathuna, "Hybrid energy storage for energy harvesting powered wireless sensor networks," in *Smart Systems Integration*, Mar. 2011, Dresden, Germany.
[32] Crossbow, "Telosb datasheet," Internet, Jul. 2013. [Online]. Available: http://www.memsic.com/userfiles/files/Datasheets/WSN/telosb_datasheet.pdf
[33] J. Polastre, R. Szewczyk, and D. Culler, "Telos: Enabling ultra-low power wireless research," in *Proc. 4th Int. Symp.IPSN*, 2005, pp. 364–369.
[34] M. D. Jovanovic and G. L. Djordjevic, "Reduced-frame TDMA protocols for wireless sensor networks," *Int. J. Commun. Syst.*, 2012, DOI: 10.1002/dac.2439.
[35] M. Snyder, V. Yu, and J. Heissler, "Two new media access control schemes for networked satellite communications," in *Proc. IEEE MILCOM Commun. Netw.-Centric Oper., Creating Inf. Force*, 2001, vol. 2, pp. 816–823.
[36] Y. Yang and T.-S. Yum, "Delay distributions of slotted aloha and CSMA," *IEEE Trans. Commun.*, vol. 51, no. 11, pp. 1846–1857, Nov. 2003.
[37] P. Levis *et al.*, "Tinyos: An operating system for sensor networks," in *Ambient Intelligent*, W. Weber, J. Rabaey, and E. Aarts, Eds. Berllin, Germany: Springer-Verlag, 2005, pp. 115–148.
[38] M. Maroti, B. Kusy, G. Simon, Á. Lédeczi *et al.*, "The flooding time synchronization protocol," in *Proc. 2nd Int. Conf. SenSys*, 2004, pp. 39–49.

Sven Jäger received the Diploma degree in computer science from Technische Universität Ilmenau, Ilmenau, Germany, in 2010. He is currently working toward the Ph.D. degree in the System and Software Engineering Group, Technische Universität Ilmenau.

He worked on industrial projects for the international civil aviation industry. His research interests include modeling and analyzing of complex systems and model-based development of simulation-based applications.

Tino Jungebloud received the Dipl.-Inf. degree from Ilmenau University of Technology, Ilmenau, Germany. He is currently working toward the Ph.D. degree with the Systems and Software Engineering Group, Ilmenau University of Technology.

He is also currently a Research Scientist and a Lead Software Engineer with Mission Level Design GmbH, Arnstadt, Germany. His research interests are in model-based system performance evaluation at mission and operation level and the implementation of standards-compliant system modeling and simulation methodologies for MLDesigner.

Ralph Maschotta received the Diploma degree in technical computer science from the University of Applied Sciences Schmalkalden, Schmalkalden, Germany, in 1999 and the Ph.D. degree from Technische Universität Ilmenau, Ilmenau, Germany, in 2008.

Since 2011, he has been a Senior Scientist and a Lecturer with the System and Software Engineering Group, Faculty of Computer Science and Automation, Technische Universität Ilmenau. His main research interests include object-oriented programming, modeling, and design of software systems, as well as image processing and image recognition in medical and industrial applications.

Armin Zimmermann received the Diploma, Ph.D., and Habilitation degrees from Technische Universität Berlin, Berlin, Germany, in 1993, 1997, and 2006, respectively.

He has been a Full Professor of system and software engineering since 2008 and the Director of the Institute for Computer and Systems Engineering since 2012 with Technische Universität Ilmenau, Ilmenau, Germany. His research interests include discrete event system modeling and performance evaluation and their tool support with embedded system applications.

Prof. Zimmermann is a member of the Industrial Automated Systems and Controls Subcommittee of the IEEE IES Technical Committee on Factory Automation. He has served as an Associate Editor of the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS since 2008.